AUTOMATED QUESTION ANSWERING:

TEMPLATE-BASED APPROACH

Eriks Sneiders

Department of Computer and Systems Sciences
Royal Institute of Technology and Stockholm University
February 2002

Abstract

The rapid growth in the development of Internet-based information systems increases the demand for natural language interfaces that are easy to set up and maintain. Unfortunately, the problem of understanding natural language queries is far from being solved. Therefore this research proposes a simpler task of matching a one-sentence-long user question to a number of question templates, which cover the knowledge domain of the information system, without in-depth understanding of the user question itself.

The research started with development of an *FAQ* (Frequently Asked Question) *answering system* that provides pre-stored answers to user questions asked in ordinary English. The language processing technique developed for FAQ retrieval does not analyze user questions. Instead, analysis is applied to FAQs in the database long before any user questions are submitted. Thus, the work of FAQ retrieval is reduced to keyword matching without understanding the questions, and the system still creates an illusion of intelligence.

Further, the research adapted the FAQ answering technique to a *question-answering interface for a structured database*, e.g., relational database. The entity-relationship model of the database is covered with an exhaustive collection of question templates – dynamic, parameterized "frequently asked questions" – that describe the entities, their attributes, and the relationships in form of natural language questions. Unlike a static FAQ, a question template contains entity slots – free space for data instances that represent the main concepts in the question. In order to answer a user question, the system finds matching question templates and data instances that fill the entity slots. The associated answer templates create the answer.

Finally, the thesis introduces a generic *model of template-based question answering* which is a summary and generalization of the features common for the above systems: they (i) split the application-specific knowledge domain into a number of question-specific knowledge domains, (ii) attach a question template, whose answer is known in advance, to each knowledge domain, and (iii) match the submitted user question to each question template within the context of its own knowledge domain.

Keywords

automated question answering, FAQ answering, question-answering system, template-based question answering, question template, natural language based interface

Acknowledgements

The writing of this thesis was greatly supported by people at the Department of Computer and Systems Sciences, a joint department between Stockholm University and the Royal Institute of Technology (Sweden).

First of all, I would like to express gratitude to my supervisor Janis Bubenko jr. who has made my doctoral studies possible, has supported me in finding my own research realm, who has taught me how to express the ideas in a scientific writing. Thanks for reading the drafts and making valuable comments.

Many thanks to Hercules Dalianis for his support during the initial and final stages of this research. Hercules clarified things in Natural Language processing for the first sketch of the licentiate thesis, and a couple of years later commented on the drafts and final version of this thesis.

I would like to thank Bengt G. Lundberg for giving me ideas and clearer vision of things in Information Retrieval and Knowledge Management.

Jacob Palme has inspired this research by actively supporting the idea of using simple means in order to create an illusion of computer intelligence. He was the supervisor of Kristoffer Larsson who made his MSc thesis within the framework of this research, who set up and maintained a question answering system on HTML. Many thanks to Kristoffer.

Thanks to my colleague Janis Stirna for remarks regarding Enterprise Modeling. My initial participation in the Elektra project and communication with people working in this project let me get better acquainted with Enterprise Modeling.

Fredrik Björck has contributed with his friendship and inspiring discussions on the theoretical and practical results of my research.

Finally, I am grateful to people at Metamatrix[1] who have stimulated the research in developing a question-answering interface of a relational database.

[1] http://www.metamatrix.se/, valid in November 2001

Table of Content

Part 1. Introduction and Overview

Part 2. Automated FAQ Answering on WWW Using Shallow Language Understanding

Part 5. Model of Template-Based Question Answering

PART 1

INTRODUCTION AND OVERVIEW

1 Overview

The amount of information available on an Internet-connected computer grows rapidly. Information management and communication technologies try to help people feel comfortable in the flood of this information, which in its turn rises the risk of getting lost in the jungle of both the information itself and the associated technologies. Meanwhile, people sitting in front of their computers expect quick solutions and quick answers to their questions. They either easily find what they are looking for or give up.

1.1 Motivation of the Research

Traditionally, question answering on the WorldWide Web (WWW) is done by the means of static lists of frequently asked questions (FAQs) and their answers. In such a list, people locate the questions of their interest and read the answers. This practice, however, has two shortcomings. First, the readers of an FAQ list do not explicitly ask any questions, therefore the information provider does not know all the variety of questions that arise. Second, finding valuable information in a long – several hundreds of entries – and chaotic FAQ list, or a number of lists, is a tedious work.

Keyword-based search, used by most search engines, is a common means of document retrieval on the Web. Many of us, however, do not think in terms of Boolean expressions made of keywords and are not used to the engine-specific syntax of such expressions. Another inconvenience of the keyword queries is the large amount of retrieved irrelevant information.

Input forms are common user interfaces for structured (e.g., SQL) databases. Input forms are convenient if they are small, but tedious if there are many input fields.

Supposedly, the most natural kind of queries posted to an information system is questions stated in ordinary human language. In a question, the user can specify exactly what he or she wants. It is more fun to talk to a computer in ordinary English. A natural language based interface does indirect interviewing of the users: in the logs of the system we read what people think when they search for information. On the web such an interface adds one more dimension – limited human language understanding – to the traditional notion of multi-media (images, sounds, animation).

1.2 Research Problem and Main Results

The overall **research problem** of this thesis is development of question-answering systems that answer one-sentence-long user questions and, thus, reduces the shortcomings of FAQ lists, keyword-based search, and large data input forms. The systems have to be inexpensive and easy to maintain, which are intuitive

requirements necessary to make the systems affordable for organizations such as, among others, university departments and small companies.

Unfortunately, the problem of understanding natural language queries is far from being solved. Therefore this research proposes a simpler task of matching a user question to a number of question templates, which cover the knowledge domain of the information system, without in-depth understanding of the user question itself.

The research started with *automated FAQ answering*. In order to give an idea of the functionality of the developed FAQ answering system, an example of asking a question follows. By using a Web-browser, the user submits his or her question to the system (Figure 1.1).

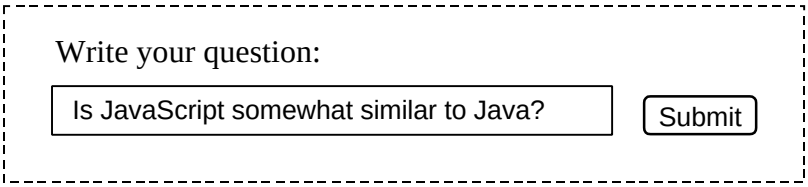

Figure 1.1 Example of a user question.

The system receives the question and searches through its database in order to find pre-stored static FAQs that correspond to the question. After one or several, if any, relevant FAQs are found, the system sends them and their answers back to the user (Figure 1.2).

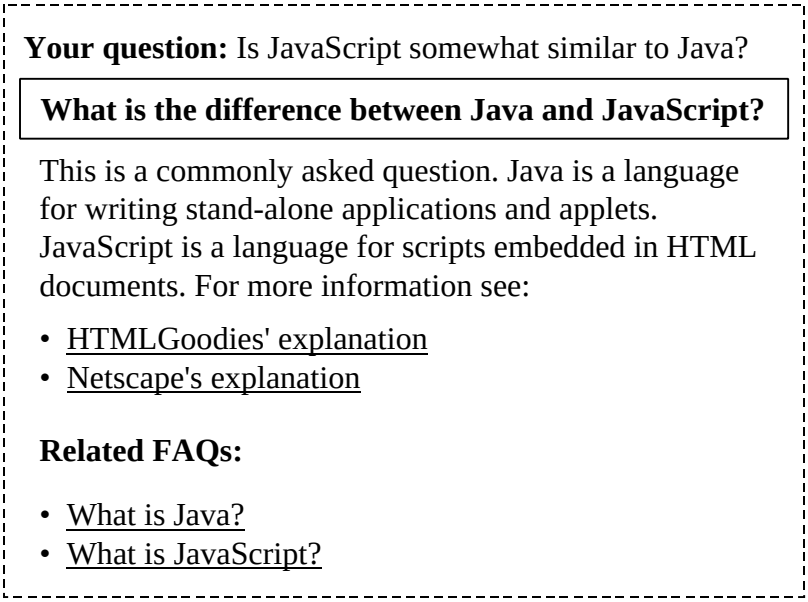

Figure 1.2 Reply to the question.

Later on, automated FAQ answering developed into a more *generic approach of template-based question answering* suitable also for building question-answering interfaces of structured (e.g., relational) databases.

The **main results** of the research are:

- the idea, data structures, algorithms, and application guidelines for an *FAQ answering system*;
- the idea, data structures, algorithms, and application guidelines for a *question-answering interface* of a structured database, a descendant of the FAQ answering system;
- the generic *model of template-based question answering* as an abstract foundation of the techniques used by the above FAQ answering system and the question-answering interface of a relational database, and possibly a number of similar systems built in the future.

1.3 Structure of the Thesis

The thesis has five parts. Each part is autonomous and should be understood without reading other parts. This does, however, lead to certain repetition of the content.

Part 1, this one, is the introduction and overview. Section 2 presents the specific research problems and their research approaches, the flow of the work and main discoveries, and finally a summary of the scientific contributions. Section 3 discusses the originality and credibility of the results, as well as outlines the guidelines for the further research.

Part 2 is the licentiate thesis "Automated FAQ Answering on WWW Using Shallow Language Understanding" [Sneiders 1999-a]. It presents the first version of the FAQ answering system working in the knowledge domain an Enterprise Modeling [Bubenko 1994] technique. Part 2 identifies the main goals, concepts, and processes of the FAQ answering system. It introduces the Prioritized Keyword Matching technique – its data structures and algorithms – used to match a user question to an FAQ entry residing in the system's database.

Part 3 consists of two publications. The paper "Automated FAQ Answering: Continued Experience with Shallow Language Understanding" [Sneiders 1999-b] was presented at the AAAI Fall Symposium on Question Answering Systems held November 5-7, 1999 in North Falmouth, Massachusetts, USA. It is a rather complete summary of the licentiate thesis. The paper introduces a few new features not covered in the licentiate thesis.

The report "Application and Maintenance Aspects of an FAQ Answering System" [Sneiders and Larsson 2001] was written after extensive testing of the FAQ answering system had been conducted. The report does not focus on the technical details of the FAQ retrieval but rather discusses the social aspects of the FAQ answering system, such as the user interface, the application domains of the system, as well as the human resources needed in order to install and maintain the system.

Part 4 is a new piece of work not published previously. It discusses the shift from the FAQ answering system to the question assistant – a question-answering interface for a structured database, further called "target database". The static FAQ entries cannot deal with a large number of data instances residing in the target database. Therefore the notion of question templates, which cover the entity-relationship (ER) model [Chen 1976] of the target database, has been introduced. Part 4 discusses the ideas behind the question assistant, its components, the semi-formal algorithms and their execution time estimations, the maintenance issues, evaluation of the performance. Part 4 should be interesting for readers looking for the practical ideas behind the design of the question assistant.

Part 5 is a new piece of work not published previously. It presents the generic model of template-based question answering – a summary and generalization of the common features of the FAQ answering system and the question assistant. Part 5 should be interesting for readers who enjoy abstractions and theoretical models.

1.4 What this Research Is and Is not

Template-based question answering has its background in the information systems' research. It is an approach to building a question-answering interface from the point of view of the information carrier. It focuses on the issue "we have information, how can we better serve it" as opposed to "we have a query, how can we answer it". The solutions of this approach are based on conceptual modeling [Loucopoulos and Zicari 1992] of information systems and use a property common for many natural language sentences and all entity-relationship models of information systems: they embody the structure "object-relationship-object-relationship…". This is *not* a research in Natural Language Processing (NLP), Artificial Intelligence (AI), or Information Retrieval. The goal of this research is *not* to confirm or deny any popular theory but rather to try out and evaluate another approach to building a question-answering interface.

2 Research Procedure

Section 1.2 states that the overall research problem of this thesis is development of inexpensive and easy to maintain question-answering systems. This section introduces more specific research stimuli and phases. Further, it presents the flow of the work and main discoveries. The section ends with a summary of the scientific contributions.

2.1 Research Phases

The specific research problems appeared successively during the course of the research. Initially the research was devoted to automated FAQ answering. Then it shifted to building a question-answering interface for a structured database. Finally, the generic model of template-based question answering was developed. The three subsequent research problems constitute three research phases.

2.1.1 Initial Phase

The idea of building an FAQ answering system on EKD [EKD 1998] – an Enterprise Modeling technique – appeared at the Department of Computer and Systems Sciences at Stockholm University and the Royal Institute of Technology when the ELEKTRA [ELEKTRA 1996] project started. The project applied EKD for solving management problems in two power supply companies. The participants of the project came from academic and business environment in different countries: Greece, Israel, France, Sweden, and UK. In order to better accumulate and distribute knowledge about EKD, it was suggested to create an FAQ answering system[2] that could answer questions asked in natural human language. Nonetheless, the proposal of such a system was not a part of the activities held within the ELEKTRA project. The idea of building the system emerged in an academic environment, and the biggest value of the system was its research outcomes – the approach, algorithms, and data structures for automated FAQ answering. Obviously, the research value of the FAQ answering system was not EKD dependent.

The first prototype of the FAQ answering system was built. Further literature studies showed that no existing text processing techniques suited the needs of the system being developed. During the *initial phase*, the research identified the requirements and alternative solutions for the FAQ answering system, applied existing solutions of computerized text processing where appropriate, and came up with original solutions were existing ones did not fit the requirements.

A notable aspect of the initial research is the fact that the very first prototype of the system was built before the literature study. The prototype showed exactly what

[2] http://ekd.dsv.su.se/faqs.htm, valid in November 2001

features the system had and what solutions were needed. The following literature study showed the strengths and weaknesses of the proposed new solutions and helped to fine-tune them.

The other instance of the FAQ answering system[3], which answered questions on HTML, was created by a MSc student under the supervision of the author of this thesis [Larsson 1999]. One of the goals of the HTML FAQ answering system was to test how another person, different from the developer of the technique and software of the system, could handle the administration of the system.

2.1.2 Second Phase

The demand for natural language interfaces on the Web goes beyond automated FAQ answering. Organizations are interested in having question-answering interfaces for structured databases, such as SQL databases. During a later stage of testing the HTML FAQ answering system, the developers of the system were contacted by Metamatrix[4], the company that had developed a database on events in Stockholm, the capital of Sweden. The database[5] is operated by Stockholm Information Service (SIS). Metamatrix wondered whether or not it would be possible to adapt the FAQ answering technique for the SIS database – an SQL database with well-structured data. The first answer was "probably no" because FAQ answering techniques deal with static FAQ entries whereas an SQL database requires dynamic queries that represent all the variety of its data instances. Nonetheless, further reasoning lead to the initial ideas concerning question templates – dynamic FAQs – having variable parameters that represent data instances in an SQL database.

During the *second research phase*, the FAQ answering technique, developed during the initial phase, was adapted for the question assistant – a question-answering interface of a relational database. The adaptation implied preserving those features that proved good and adding new ones required by the new conditions.

2.1.3 Third Phase

The AAAI Fall Symposium on Question Answering Systems [Chaudhri and Fikes 1999], held November 5-7, 1999 in North Falmouth, Massachusetts, USA, was an event important for this research. The conference had a specific focus on automated question answering as opposed to more general natural language processing. Practitioners were especially welcome. During the opening speech, the co-chair Vinay K. Chaudhri talked about different approaches and, among others, mentioned

---

[3] http://www.dsv.su.se/html/, valid in November 2001

[4] http://www.metamatrix.se/, valid in November 2001

[5] http://www.stockholmtown.com/events/9/index.asp, valid in November 2001

template-based question answering. The concept "template-based question answering" had first appeared within the framework of this research. At that time, the concept had not much public content attached to it. Template-based question answering was attributed mostly to Ask Jeeves[6] – a company that operated and never published its own proprietary technology.

The paper "Automated FAQ Answering: Continued Experience with Shallow Language Understanding" (included in Part 3 of this thesis) was presented at the conference. The idea of question templates having variable parameters (see Section 2.1.2) was formulated shortly before the conference. The more generic notion of template-based question answering, acquired at the conference, allowed bringing the FAQ entries and the question templates under a common umbrella.

During the *third research phase*, a few similar systems were observed and analyzed in order to distinguish, explain, and summarize their common features. As the result of the analysis, the model of template-based question answering was built.

2.2 Course of the Research

The three research approaches and main results have split the research into three logical parts. This section gives an insight into the course and discoveries of these parts.

2.2.1 Automated FAQ Answering

Unlike traditional AI question-answering systems that generate new answers, FAQ answering systems match the submitted user question to pre-stored FAQs in the database and retrieve those FAQ-answer pairs that are semantically close to the question (Figure 2.1). In order to accomplish this task, a system has an appropriate FAQ retrieval technique.

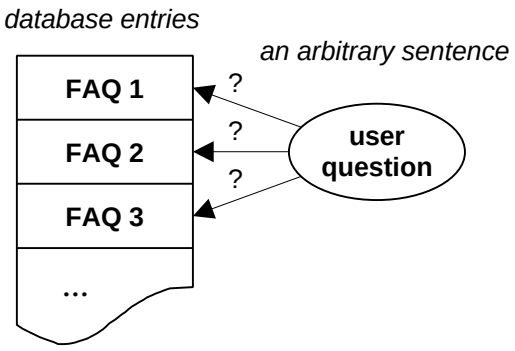

Figure 2.1 Matching a user question to FAQs.

The statistical methods of Information Retrieval determine similarity between free text documents by counting frequency of common terms in the documents. These methods were designed to process large pieces of text and are not appropriate for FAQ answering: single sentences are too short for calculation of term frequency.

On the other hand, the systems that do semantic natural language processing are known being difficult to build and maintain. They are expensive. They require

[6] http://www.askjeeves.com/, valid in November 2001

qualified personnel with specific skills in NLP, something that many organizations do not have. The following excerpt from [Mearian 2001] gives us a hint about what it could take to acquire an "intelligent" question-answering interface:

> "Last month, Schwab completed a nine-month installation of a natural-language search technology on its Web site […]. Schwab said the technology allows its 7.5 million end users to type simple or complex questions and receive answers from anywhere on the site in one step, bypassing the need for other, more complex navigation tools.
>
> Schwab.com chose One Step, an application created by start-up vendor Iphrase Technologies Inc. in Cambridge, Mass. […]
>
> Prices for One Step run from $250,000 for an annual subscription to $700,000 for the entire software suite. […]
>
> The installation of One Step included about three months spent on peer evaluation within the company, allowing employees to test the technology before subjecting the public to it."

There are many organizations that cannot afford a system like One Step.

The *Prioritized Keyword Matching* technique was developed by this research as a simple means of matching a user question to an FAQ entry. The idea of the technique is based on the assumption that there are three main types of words in a sentence within a certain context in a certain subject: (i) required keywords which convey the essence of the sentence, (ii) optional keywords which convey the nuances of the sentence and can be omitted without changing the essence, and (iii) generally "irrelevant" words, like "a", "the", "is", etc., that are too common in ordinary language or in the subject. In Part 2 of this thesis – the licentiate thesis – required and optional keywords are called primary and secondary keywords respectively. The names were changed as the third type of keywords – forbidden keywords – appeared.

All FAQ entries in the system's database have their required and optional keywords, as well as some other data specified. After the user submits a question, the system iterates through the FAQ entries and applies the Prioritized Keyword Matching algorithm to each entry in order to determine whether or not the FAQ is semantically close to the user question. Complex FAQs are represented by a number of auxiliary entries – one auxiliary entry for each alternative form of the FAQ.

The FAQ entries are created manually. This research does not propose miraculous solutions of the long lasting problems of Artificial Intelligence. Intelligent decision making during the process of creating FAQ entries takes human reasoning because today's technologies cannot communicate the meaning of a natural language sentence from the human mind to the computer without any manual work done at

some point of the development of the system itself or its imported components.

The simple technique proved effective: it is easy to create and maintain the FAQ entries, and the quality of FAQ retrieval is fairly good.

2.2.2 Question Assistant for a Structured Database

A static FAQ embraces one or several, usually no more than two, generic concepts and the relationships between them. If we want to include more specific instances of the concepts, we have to create a separate FAQ entry for each combination of the instances. This is not a smart solution if the number of the specific instances is large.

A *question template* is a dynamic FAQ as opposed to the traditional static FAQ. It is a question having entity slots – free space for data instances that belong to the main concepts of the question. For example, "When does `<artist>` perform in `<place>`?" is a question template where `<artist>` and `<place>` are the entity slots. If we fill these slots with data instances that belong to the corresponding entities of the target database, we get an ordinary question, e.g., "When does Pavarotti perform in Globen?"

An ER model is a graphic way of displaying real world concepts, their relationships, and attributes. A written natural language sentence is another way of displaying real world concepts, their relationships, and attributes. By a process called conceptualization, information is transformed into sentences, sentences are transformed into elementary sentences, and elementary sentences into object-role pairs. A conceptual schema describes which elementary sentences may enter and reside in the information system [Nijssen 1977]. Because an ER model, which is a conceptual schema, embodies a set of elementary sentences, we can map a natural language question into an ER model. Figure 2.2 shows an example of such a mapping: each concept and relationship expressed in the question has a counterpart in the ER model.

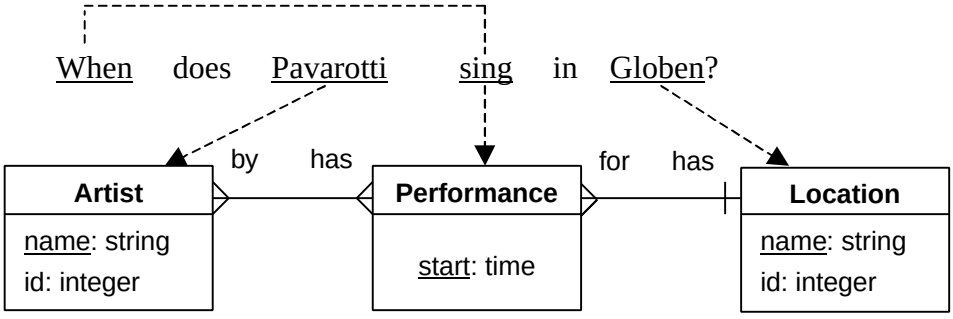

Figure 2.2 Natural language sentence mapped into an ER model.

An ER model represents a knowledge domain. Because the ER model of a stable information system is rather static, we can cover the model with a number of question templates – "frequently asked questions" – that represent the entities, their

relationships, and attributes in the form of natural language sentences. An exhaustive collection of question templates meets the majority of conceivable user questions that the information system is capable to answer.

The question template's "answer" is created by the help of a *database query template* – a formal database query having free entity slots for data instances, primarily primary keys or their equivalents. After the entity slots are filled, the template becomes an ordinary executable database query.

Answering a user question takes five steps:

1. When a user submits a question, the system identifies the data instances and their corresponding entities referred to in the question. Special algorithms are used for this purpose.
2. The system matches the submitted user question to the question templates and retrieves one or several, if any, question templates that are (i) semantically close to the user question – Prioritized Keyword Matching is used here – and (ii) refer to the same entities as the user question.
3. The system presents the matching question templates and data instances to the user as interpretations of the original user question.
4. The user selects an interpretation and resubmits it as a refinement of the original question.
5. The system answers the interpretation.

Most interesting, from the point of view of this research, are the algorithms in (1) and (2). Their execution time is equivalent to that of popular sorting algorithms – $\boldsymbol{O}$ ($N \log(N)$) – where N is the number of data instances in the target database.

2.2.3 Towards the Model of Template-Based Question Answering

The FAQ answering system and the question assistant for a structured database have common main principles: they *cover the knowledge domain with a number of FAQs / question templates*, whose answers are known in advance, and match the submitted user question to these FAQs / question templates.

Another common feature is use of a *multiple lexicon*. The question templates, once created or updated, are static. The keywords are known long before any user questions are asked. Each question template has a small lexicon that implements one function – it identifies mutually exchangeable words (synonyms and their grammatical forms) for every keyword within the context of a given question template. The system operates a multiple lexicon assembled from numerous independent small lexicons, each attached to its own question template.

In a broader sense, the idea of multiple lexicon suggests splitting the application-specific knowledge domain into a number of question-template-specific knowledge domains. The multiple lexicon solves a long lasting problem of NLP –

disambiguation of the meanings of words in the context of a short expression. The lexicon consists of a number of small, analogous, autonomous units which operate 5-10, or even fewer, concepts in the context of the question in one question template. On the contrary, the traditional NLP lexicon operates thousands of concepts in the context of the application-specific knowledge domain.

The model of template-based question answering comprises two views on the approach. The first view shows the approach in the context of a knowledge domain: how the knowledge is transformed into question templates, how and why question templates acquire their main components and features. The second view shows the constituents of the template-based question-answering approach – the generic components of the system and the user query.

2.3 Summary of the Scientific Contributions

The *Prioritized Keyword Matching* technique was initially developed for automated FAQ answering. The thesis presents the data structures and algorithms of the technique, evaluation of its performance, as well as the guidelines how to maintain the system that makes use of the technique.

The question assistant – a question-answering interface for a structured (e.g., relational) database – covers the ER model of the database by a number of *question templates*. An exhaustive collection of question templates meets the majority of conceivable user questions that the information system is capable to answer. The thesis presents the data structures and algorithms of the question assistant, the estimated execution time of the algorithms, preliminary evaluation of the performance, as well as the maintenance guidelines.

The *multiple lexicon*, assembled of a number of independent small lexicons where each of them is attached to its own question template, solves a long lasting problem of NLP – disambiguation of the meanings of words in the context of a short expression, i.e., the question in one question template.

The *model of template-based question answering* describes the essential guidelines and concepts of the new approach to building question-answering systems. The model is not fastened to any particular natural language processing technique, which enhanced its theoretical value.

3 Originality and Credibility of the Research Results, Further Research

Two important qualities of a Ph.D. thesis are originality and credibility of the research results. This section discusses both qualities, as well as the further research.

3.1 Originality of the Results

Traditionally, processing of free text user input has been the endeavor of NLP and Information Retrieval. This is *not* a research in NLP or Information Retrieval. This research presents an original view on building question-answering systems, a view that stems from conceptual modeling of information systems.

A number of FAQ answering systems have been built before this research started (see Section 1.4 in Part 2 of this thesis). They use different FAQ retrieval techniques. To our knowledge, however, the Prioritized Keyword Matching technique used for FAQ retrieval is an original contribution of this research.

The originality of this research may be undermined by the large number of similarities between the techniques described in this thesis and those used by Ask Jeeves (see Section 7.4 in Part 4). Ask Jeeves and the author of this thesis did their work independently. Ask Jeeves is a company which has developed a proprietary technology, whereas this is a post-graduate research accompanied with a number of publications. Despite the similarities, the author of this thesis denies any possible allegations in plagiarism.

The idea of question templates is not an original contribution of this research; Ask Jeeves uses question templates. Not original is also the suggestion to build user interfaces based on the ER model of the information system (see Section 7.3 in Part 4). Nonetheless, the approach of mapping question templates into an ER model in order to create a question-answering interface is novel.

The idea of multiple lexicon is original. It appeared as the result of the analysis of the FAQ entries used for Prioritized Keyword Matching.

The name “template-based question answering” is not an invention of this research. It comes from the 1999 AAAI Fall Symposium on Question Answering Systems. Still, the model of template-based question answering is a novel contribution of this research. It was created as a summary and generalization of the features common for the systems developed by this research and some other systems (see Section 3.5 in Part 5).

One may wonder why the approach of template-based question answering was not proposed earlier, say, ten years ago. Most probably the reason was lack of demand for question answering systems before the Internet made databases accessible to a

huge audience of users.

3.2 Credibility of the Results

A new question-answering approach – template-based question answering – has been developed by this research. In order to prove the usefulness of template-based question answering, we highlight its advantages and show that, under certain conditions, it yields a better utility than other question-answering approaches. The thesis does not claim that the proposed principles, data structures, and algorithms are absolutely the best solutions ever possible.

The three main criteria used to evaluate the significance and credibility of the results of the research are (i) empirical experience and measurements of the performance of the implemented systems, (ii) links to related research and systems, and (iii) intelligent analysis and generalization of the features of the implemented systems.

Empirical Experience

The capabilities of the question-answering systems developed by this research have been tested by measuring the recall (the system's ability to retrieve all the relevant documents) and precision (the system's ability to retrieve only relevant documents) of two FAQ answering systems (see Section 5 in Part 2 and Section 6 in Part 4 of this thesis). The figures are good in comparison to the average recall and precision of Information Retrieval systems. Nonetheless, the figures are subjective with respect to the administrators of the systems who manually created the FAQ entries.

The thesis claims that the template-based approach makes question-answering systems easier to deploy and maintain in comparison with traditional NLP systems. With respect to automated FAQ answering, only two people – the author of this thesis and a MSc student who set up the HTML FAQ answering system – have tested the claim. Two people are not enough to sustain the claim by the means of an empirical study. The claim is based primarily on the analysis of the features of the model of template-based question answering.

Salut – a European Union 5th Framework research project which maintains a Web-site about eating disorders[7] – has started deployment of equivalent FAQ answering systems in English and Swedish. There are plans for French and Spanish. Hopefully, the Salut project will bring more empirical evidence of how easy or difficult the proposed question-answering techniques are.

Links to Related Research and Systems

The model of template-based question answering uses results of decades long research in conceptual modeling of information systems, including the research on

[7] http://www.salut.nu/, valid in November 2001

the links between a conceptual model and the human language done as early as in 1970-ies and 1980-ies (see Section 1.2 in Part 4 and Section 2.1 in Part 5). The possibility to cover a knowledge domain with an exhaustive number of question templates is determined by the possibility to create a reasonably complete conceptual model of the knowledge domain.

The similarities between Ask Jeeves and the question assistant suggest that there must be strong rationale behind the design of both systems, as two independent researches have achieved roughly the same results (see Section 7.4 in Part 4).

Analysis of the Features of the Implemented Systems

The generic model of template-based question answering, introduced in Part 5 of this thesis, was created as a summary and generalization of the features common for the systems developed by this research and some other systems. There are not enough empirical studies done in order to prove the utility of all the variety of the features of the generic model of template-based question answering. The significance of the model has been established by intelligent analysis of its features (see Section 2.4 in Part 5).

3.3 Further Research

In the end of Parts 2, 4, and 5 there are sections that describe the further research relevant for the particular part. The following guidelines illustrate the possible directions of the further research in general.

An interesting and essential research direction is to make the question-answering systems understand human languages other than English, Swedish for example. The current implementation of Prioritized Keyword Matching may experience difficulties if it gets compound words, which are so common in Swedish, in the user input unless splitting of compound words into their constituents is resolved. The techniques that process the text of the data instances that fill entity slots of question templates are going to experience similar problems.

It is not clear yet how NLP techniques would work in the settings of template-based question answering. We may want to upgrade Prioritized Keyword Matching, which does not perform semantic analysis of user questions, with a more powerful sentence matching technique.

The current version of the question assistant works as a question-answering interface for a relational database. Nonetheless, the template-based question answering approach relies on the properties of the conceptual model of an information system rather than the properties of the tabular structure of a relational database. Further investigation could show how to adjust the question assistant to question answering from, say, an object-oriented database.

Automated tools that help to extract question templates form the conceptual model

of the information systems, as well as tool that help to create autonomous units of a multiple lexicon would be rather helpful.

PART 2

AUTOMATED FAQ ANSWERING ON WWW USING SHALLOW LANGUAGE UNDERSTANDING

Preface for Part 2

Part 2 comprises the licentiate thesis “Automated FAQ Answering on WWW Using Shallow Language Understanding” [Sneiders 1999-a].

The layout and pictures in this edition are adjusted to the new format and are different from those in the original publication. Internet addresses in the references are updated.

The readers interested in a good **summary rather than a detailed description** of the research in automated FAQ answering are recommended to skip Part 2 and read the paper “Automated FAQ Answering: Continued Experience with Shallow Language Understanding” in Part 3.

1 Introduction

The subject of this research is development of an evolving automated FAQ (frequently asked question) answering system that provides pre-stored answers to users' questions asked in ordinary English. The World Wide Web (WWW) is the medium of communication between the system and its users.

The domain of the questions is EKD (Enterprise Knowledge Development) [EKD 1998][8] – an Enterprise Modelling technique [Bubenko 1994]. Therefore the system is called the EKD Question Answering (QA) System. Nonetheless, the requirements of the system, its architecture and functionality, as well as the methods of its development, use, and maintenance are general and do not depend on EKD as the subject.

Unlike traditional Information Retrieval systems [Salton and McGill 1983; Salton 1989] aimed primarily at indexing and retrieval of external documents, the EKD QA System deals with creation, maintenance, indexing, and retrieval of internal documents, i.e., FAQs and their answers. These FAQs do not exist outside the system, therefore both the creation and retrieval are equally important. Hence, there are similarities between the EKD QA System and traditional Information Retrieval systems, but the accents are different. The functionality, maintenance, algorithms, and data structures of the EKD QA System are different from those described in [Salton and McGill 1983; Salton 1989].

Unlike some other natural language question answering systems [Hammond et al. 1995; Grosz et al. 1986] that perform syntactical and lexical analysis of user queries, the EKD QA System does not analyze the queries during FAQ retrieval process – intelligent keyword matching proved sufficient.

1.1 WWW

The recent boom of the Internet has established a new environment for information exchange and has substantially broadened capabilities of communication and availability of information systems. In 1969, the first trial of a packet-switching network was conducted. The four-node network connected the University of California Los Angeles, Stanford Research Institute, the University of California Santa Barbara, and the University of Utah [Cerf 1993]. During the last five years, the estimated number of computers connected to the Internet has grown 28 times and has reached 36.74 million in July 1998 [ISC 2001].

1.1.1 Why is the Web so Popular?

On November 12, 1990 "WorldWideWeb: Proposal for a HyperText Project" was

[8] Also http://ekd.dsv.su.se/, valid in November 2001

made at the European Laboratory for Particle Physics (CERN). “The WWW project was originally developed to provide a distributed hypermedia system which could easily access – from any desktop computer – information spread across the world. The Web includes standard formats for text, graphics, sound, and video which can be indexed easily and searched by all networked machines.” [Gromov 1995-2001] Today, WWW is one of the most extensively used Internet services. If available, brief information on WWW is preferred to paper printed references because in most cases Web-based information is kept up to date and available for any computer connected to the Internet unless special restrictions are applied. Internet search engines do indexing of the Web and help to find information. Multi- and hyper-media technologies enrich the forms of presentation of information. The number of WWW servers has grown from about 50 in January 1993 [Cailliau 1995-2001] to roughly 2 595 000 in July and 3 359 000 in October 1998[9]. Figure 1.1 illustrates the increase of the number of WWW servers during the last years.

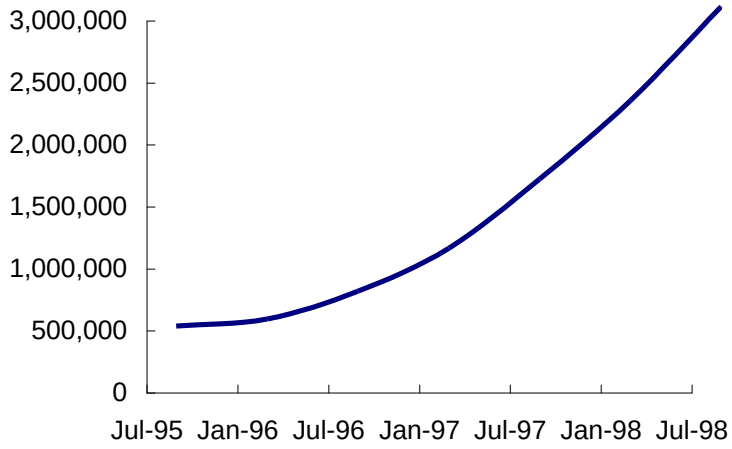

Figure 1.1 Growth of the number of WWW servers during the last years.

WWW is based on the computer network client-server architecture, where the client is a Web browser, and the server is an HTTP (HyperText Transfer Protocol) server. HTML (HyperText Markup Language) is a shell that encapsulates multimedia information stored on websites (server computers) and presented to users. HTTP is the communication protocol between a browser and a server. Both HTML and HTTP were initially designed to disseminate information in one direction – from the server (information supplier) to the browser (information consumer) – with limited possibilities for the browser to communicate backwards. Today, however, HTML and HTTP support easy data exchange in both directions, which enables even elaborate remote database inquiries. Since the WWW user

[9] http://www.netcraft.com/survey/, the figures for 1998 were not available in November 2001

interface and communication protocol are standardized and operating system independent all over the world, Web browsers are convenient to use as user interface tools for Internet-based applications. Examples of such applications are Basic Support for Cooperative Work[10] and Web for Groups[11]. The EKD QA System developed within the scope of this research is another such application.

1.1.2 Problems with FAQ Lists

Often websites provide information support to the products, services and other activities that different organizations advertise. A distinguishing feature of such a websites is a collection of FAQs, i.e., frequently asked questions, where the information supplier tries to answer in advance the typical questions that the readers may have. Usually, FAQs are intended to solve certain problems. FAQs are a significant complement to the supplied information, which deepens people's understanding of this information. Traditionally FAQs are organized as an ordinary list. This approach has several deficiencies:

- *An FAQ user is not given a chance to ask any questions.* Instead, the user is forced to scan through a long list (or several lists) of various questions in order to find a question, similar to the user's own question, that may even do not exist in that list. Not been able to solve his or her problem, the user reports it by e-mail usually only if he or she is desperate.
- *The information supplier does not know the actual questions that arise.* Rather, the information supplier answers possible questions in advance. However, these possible questions do not always satisfy the users' needs. The author of this paper had bad experience when he checked a long and chaotic list of technical support questions on the Borland Delphi website[12]. Unfortunately, he could not find a solution to his specific problem. He was not given an opportunity to ask any question either. The Delphi maintainers will never be able to analyze his needs and include them in the technical support question list.
- *FAQs in a list may be poorly organized.* If the number of FAQs is large and their order chaotic, then there are two options to navigate through the list: (1) to read all the questions or (2) to search for keywords by free text substring search facility. It is, however, not possible to use substring search if the list is spread over several HTML documents. It is an advantage if the FAQs in the list are semantically grouped. But even in this case the grouping may be ambiguous, and a user may not know where exactly to look for a particular question.
- *There may be several FAQs in a list that answer the user's question.* Several

[10] http://bscw.gmd.de/, valid in November 2001

[11] The server was not accessible in November 2001

[12] http://community.borland.com/delphi/, valid in November 2001, select "TI's"

more FAQs may be related to the question. If the list is not well structured, a user has to scan through the whole list in order not to loose possibly valuable information.

- *An FAQ list may be too long, sometimes scattered over several HTML documents.* It is difficult to look for a small piece of information in a large text mass.

These deficiencies pertain to FAQ lists whatever medium carries them – WWW, compact disc, paper, etc. WWW, however, is special because it is a rapidly growing medium, convenient for asynchronous communication, and popular in the business and academic environments. Therefore this research focuses on automated FAQ answering on WWW.

1.2 EKD

This section briefly introduces us with EKD – the knowledge domain of the EKD QA System – and the role of EKD in this research.

1.2.1 What is EKD?

EKD is an Enterprise Modelling technique. Enterprise Modelling is a general business planning methodology which stems from the traditional Requirements Engineering and extends its initial capabilities. Basic ideas related to Enterprise Modelling were introduced in the beginning of the eighties and refined in the late eighties. The significant contribution here was the notion of considering intentional components of a requirements specification, e.g., the goals and intentions of a business, in addition to traditional component types such as entities, relationships, and processes [EKD 1998, p. 9].

Enterprise Modeling helps to make tacit knowledge explicit. By focusing on different views of an enterprise, the Enterprise Modelling methodology tackles ill-defined problems that typically occur in the development of an enterprise. It can support issues such as business strategy planning, definition and analysis of business concepts and business rules, continuous monitoring of business activities, reengineering of business processes, restructuring of the enterprise, design of an information system or product, human resource planning, etc. An enterprise can be seen as an organization, a part of an organization, or an activity we want to study and describe. It can be an enterprise as it is today, or it can be a vision we have about the future of it.

We distinguish two key aspects of using EKD as an Enterprise Modelling technique: the modelling process and the result of this process. The *result* is an Enterprise Model – a system of knowledge represented by a set of diagrams. It gives better-structured and clearer understanding of the enterprise's goals, business processes, concepts, rules and resources. It helps to better control a complex

situation. We can see where we are, where we want to be, and how to get there. *Modelling* of an enterprise is a process of defining and analyzing its goals, business processes, concepts, rules and resources. Modelling is like putting pieces of a puzzle together to get the whole picture. The modelling process considerably improves understanding of the enterprise, reveals hidden objectives, relationships, threats, helps to find solutions to difficult problems, improves team communication, helps to reach consensus about vague and problematic issues. While modelling their organization, people gain insight into its nature. Enterprise Modelling is beneficial in situations where there are conflicting views between business units or where people lack a common language in communication and misunderstand each other.

George P. Huber in [Huber 1984, pp. 931, 934] describes post-industrial society as one which "will be characterized by more and increasing knowledge, more and increasing complexity, and more and increasing turbulence." Therefore, "in order to enhance their decision making processes, post-industrial organizations [i.e., organizations in post-industrial society] will adopt on a widespread basis three design features: (1) advanced communication and computing technologies, (2) improved decision-group technologies and structures, and (3) 'decision-process management'". Enterprise Modelling in general, and EKD in particular, are technologies to support group decision-making process.

Enterprise Modelling has been applied in a number of companies such as British Aerospace (UK), Ceselsa (Spain), Telia, Ericsson, Sweden Post, Postgirot, Vattenfall (Sweden), Public Power Company, and National Bank of Greece (Greece). The business settings in which the method was applied include business process improvement, business strategy planning, and information system requirement definition [ELEKTRA 1996, p. II-7].

1.2.2 Role of EKD in this Research

EKD is an Enterprise Modelling technique applied in industry. Not only Enterprise Modelling experts work with EKD. A large share of responsibility for successful development of a good Enterprise Model lies on the experts in a particular business knowledge domain.

EKD is a versatile and rather abstract technique. Therefore a common problem is that the business experts who are offered EKD to use in modelling their business do not fully understand this technique, its power and potential.

EKD lacks an officially published source of explanatory information. To the moment of the beginning of this research, business experts were introduced to EKD mainly by personal contacts. Nonetheless, the developers and users of EKD and its predecessor F^3 (From Fuzzy to Formal) [F3 1994] are spread over different organizations, cities, and countries. Personal contacts are not always possible because of the distance and busyness of the people. In order to improve

dissemination of the information on EKD, the EKD website[13] has been created at the Department of Computer and Systems Sciences, a joint department between Stockholm University and the Royal Institute of Technology (Sweden). Ideally, it should contain all the information the EKD users may need in an easy to understand form. In practice, this is never the case. The information is never complete. Some pieces of information may be missing. Often a user cannot find what he or she needs, or cannot highlight a particular view of the information, or has problems with practical use of EKD, etc. Then the user asks questions.

This research aims at improving one aspect of learning EKD – automated answering of questions about EKD performed by the EKD QA System.

We should not overestimate the role of EKD in this research. The subject of the research is automated FAQ answering where EKD happened to be the topic of FAQs. The requirements of the EKD QA System, its architecture and functionality, as well as the methods of its development, use and maintenance are general and do not depend on the knowledge domain of the system.

1.3 Research Problem of this Thesis

People sitting in front of their computers expect quick solutions. People browsing a website want to get quick answers to their questions. **The research problem of this thesis is development of an automated FAQ answering system** that avoids the problems typical for traditional FAQ lists, introduced in Section 1.1.2. The EKD QA System is a Web-based information system aimed at supporting asynchronous communication between the EKD users and developers by automated question answering. The users can ask their questions in ordinary English, thereby the system adds one more dimension – limited human language understanding, to the traditional notion of multi-media technology (images, sounds, animation) on WWW.

A user can:

- ask a natural language question so that the system finds in its database and retrieves corresponding FAQs, if any;
- search by keywords through all the FAQs in the database; and
- browse all the FAQs in the database.

In order to give an idea of the system's functionality, an example of asking a question follows. By using a Web-browser, the user submits his or her question to the system (Figure 1.2).

[13] http://ekd.dsv.su.se/, valid in November 2001

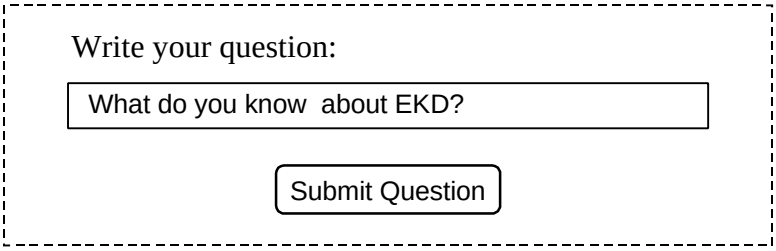

Figure 1.2 Example of a user's question to the EKD QA System.

The system receives the question and searches through its database in order to find one or several pre-stored FAQs that correspond closely enough to the user's question. Naturally, the system must recognize differently formulated questions that carry similar meanings. After the FAQs are found, the system sends them and their answers back to the user. The reply is showed in Figure 1.3.

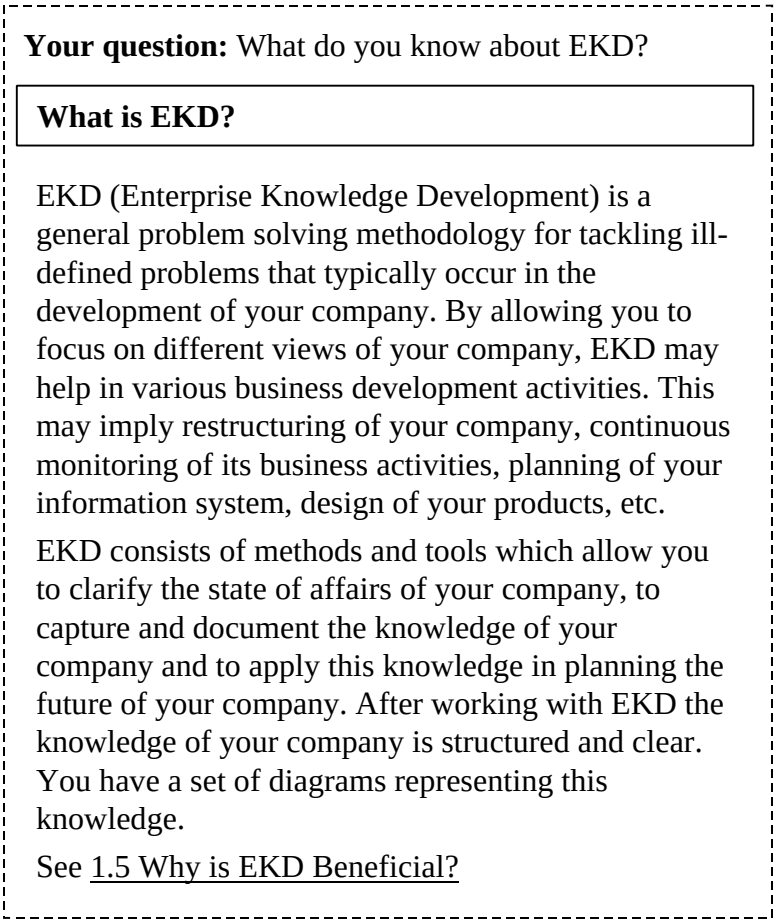

Figure 1.3 Reply to the question.

The figure above presents a close answer to the question; the system suggests related FAQs as well. Users are encouraged to ask any questions they have about EKD and Enterprise Modelling. Therefore in this thesis the concept of FAQ (frequently asked question) is extended to any question potentially interesting to the

users of EKD and included into the database of the EKD QA System.

We can distinguish the following roles of an automated FAQ answering system in the community of its users:

- *Means of information acquisition.* The system's natural language based user interface lets people formulate their problems and submit them as questions to the system that records and tries to answer these questions.
- *Form of organizational memory.* J. P. Walsh and G. R. Ungson refer to organizational memory as storage bins containing information about past decision stimuli and responses [Walsh and Ungson 1991, pp. 61, 63]. The FAQ answering system contains:
  - Identified problems (stimuli mentioned above). Each FAQ identifies a problem that has appeared within the community of the users of the system.
  - Solutions to these problems (responses mentioned above). Each FAQ has an answer that explains the solution to the problem expressed in the FAQ.
- *Means of information retrieval.* The system retrieves FAQs and their answers upon request expressed in natural human language.

"Ackerman and Malone's innovative ideas about growing an 'answer garden' in an organization testify to the promise of information technology as a means to help retain and retrieve past experience in organizations." [Walsh and Ungson 1991, p. 70] This was said about information retrieval within an organization in general, but sounds as if said about an automated FAQ answering system – an answer garden that retains and retrieves past experiences.

The lion's share of this research is devoted to the last role of an FAQ answering system – means of information retrieval, especially to development of a natural language processing and FAQ retrieval technique that enables question answering. This research does not cover the social aspects of the EKD QA System, the impact that the system makes on the community of the EKD users. Main realms of the research are the following:

- *Definition of the principles, features, concepts, and functionality* of the EKD QA System is done first.
- *The natural language processing and FAQ retrieval technique* developed within the scope of this research and used by the EKD QA System represents the biggest contribution of this research. Automated FAQ answering implies two kinds of activities: creation of an FAQ index and retrieval of FAQs from the database by using this index. Both activities are not equally well covered in this thesis. The main focus of this research is on the retrieval of FAQs and the data structure of the associated index. The algorithm and data structure were developed within the scope of this research. Optimization of the FAQ indexing

process is not covered in this thesis.

- *Creation and maintenance of the FAQs* begins when the initial set of FAQs is made before the system is put into operation; otherwise the system is useless. As the system is operated, the FAQ set must be updated considering the asked questions. Monitoring of the system's performance is necessary in order to evaluate and improve the language processing technique. The process of creation and maintenance of the FAQs is discussed within the extent necessary to get the EKD QA System running. Profound analysis and optimization of this process are not covered by this research.
- *Evaluation* of the EKD QA System deals primarily with two independent key factors that influence the system's ability to answer questions: quality of the language processing technique (as a question is asked, the system must find the corresponding FAQ) and completeness of the FAQ set (the corresponding FAQ must exist in the database). Both factors are evaluated separately.

1.4 Related Approaches to Automated FAQ Answering

Unlike Artificial Intelligence question answering systems that focus on generation of new answers, FAQ answering systems retrieve existing answers from their databases. Auto-FAQ [Whitehead 1995] and FAQ Finder [Hammond et al. 1995; Burke et al. 1997-a; Burke et al. 1997-b] are two representative systems aimed at automating navigation through FAQ sets. They have three common core features:

1. The systems use a natural language based interface – a user asks his or her question in ordinary English.
2. The systems answer it by one or several pre-stored related questions and their answers, if any.
3. Both systems interact with their users through WWW (initially FAQ Finder did not have a Web-based user interface).

Auto-FAQ has the following principles of FAQ management and retrieval:

4. The system maintains its own FAQ set, it does not navigate through external FAQ lists as the information source to answer users' questions.
5. Shallow language understanding is used in order to process a user's question and to find corresponding FAQs, if any, in the database.

   What does shallow language understanding mean? This notion is intuitive. By using shallow language understanding, the system does not comprehend a user's question. It matches the question to FAQ entries in the database. The matching is based on keyword comparison; the system does not identify semantic concepts in the question. Limited lexical analysis, however, is welcome to enhance the language processing.

6. If the question cannot be answered immediately, it is stored to be answered later by experts.

FAQ Finder is designed to improve navigation through already existing Usenet newsgroup FAQ collections organized as question-answer pairs in distributed text files. “While FAQ Finder is in some sense a browser, […] actual goal is to free users from doing any browsing at all”, the system will do it for them [Hammond et al. 1995, p. 81]. FAQ Finder receives a natural language question, identifies the files relevant for the question, and then matches the question to the segments of text that are used to organize the files themselves (e.g., questions, section headings, keywords, etc.). FAQ Finder is constructed with a combination of three technologies:

- Statistical information retrieval is used to select FAQ files that cover the topic of a particular asked question.
- Syntactic parsing is used to construct a simple parse tree and identify the primary verb and noun phrases in a question.
- Semantic concept matching, through the use of the WORDNET network of lexical semantics, is used to select possible matches between the query and target questions in the FAQ files.

Auto-FAQ and FAQ Finder represent two approaches to automated FAQ answering:

- Auto-FAQ maintains its own FAQ set; no automated indexing of an external FAQ source is performed. The natural language processing technique uses shallow language understanding based on keyword comparison, no semantic analysis of user queries is done.
- FAQ Finder does automated indexing and navigation through an external FAQ source. This approach requires limited semantic analysis of user queries.

Within the scope of this research, the author has used and developed some ideas implemented in Auto-FAQ. Auto-FAQ and the EKD QA System share the features numbered 1-6 above. Nonetheless, since the implementation details are not published in [Whitehead 1995], the matching algorithm and data concepts introduced in this thesis are considered being new. After a version of Auto-FAQ had been built, the development of the system stopped (according to personal communication with S. D. Whitehead).

There have been developed a number of natural language question answering systems. [Grosz et al. 1986] presents three question answering systems; one of them was developed as early as in the middle of seventies. Recently, some natural language processing facilities have appeared in office software and on WWW. Microsoft Office Assistant (also called “help”), which works with Microsoft products such as Word and PowerPoint 97, allows asking natural language

questions. Considering the spelling and grammar checking facilities of the Microsoft Office applications, there are grounds to say that Assistant uses semantic analysis of user queries. AltaVista[14] – one of the biggest Internet search engines – has a rather good natural language question answering ability. The quality of this ability of another engine – Excite Travel[15], is currently doubtful (November 1998). If a user asks the question "How can I get to Sweden?", the engine retrieves information about travel destinations in Sweden. If, however, a user asks "What is the meaning of life?", the reply suggests traveling to North America and some tropical countries without any explanation why. Although the language processing facilities of the Internet search engines seem being commercial secrets, the database size (millions of entries) makes us think that limited semantic analysis of user queries is used in order to select a relevant subset of database entries.

A survey of publications and existing systems suggests that the approach of shallow language understanding in automated question answering is not typical.

1.5 Contributions and Structure of this Thesis

1.5.1 What this Research is and is not

The purpose of this subsection is to advise the reader to adjust his or her expectations of the EKD QA System to what the system actually is. The EKD QA System is not a magic computerized tool that reads EKD manuals in order to find out what EKD is, performs Enterprise Modelling in order to gain experience, talks to people in order to find out their problems, creates FAQs, and, finally, answers users' questions. All the "magic" of the EKD QA System is that its database is a container of the FAQs and their answers, and the system performs automated retrieval of the FAQs and their answers upon a user's request expressed in ordinary English. The other tasks – creation, maintenance, and indexing of the FAQs – involve human reasoning. The main contribution of this research is the development of a simple technique which can ensure high – higher than usual for this kind of systems – quality of automated FAQ retrieval. This research does *not* attempt to eliminate human participation and reasoning during the administration of the system. This research does not deal with optimization of the system's administration either.

The EKD QA System is an FAQ answering system that maintains its own FAQ set. Its approach to automated FAQ answering was designed particularly for this kind of systems. The approach should *not* be misleadingly generalized; it should *not* be misapplied to the systems that perform browsing through an external FAQ source.

---

[14] http://www.altavista.com/, valid in November 2001

[15] http://www.city.net/, has a different functionality in November 2001

1.5.2 Structure of this Thesis

This thesis has two central themes: (1) design and functionality of an FAQ answering system and (2) natural language processing and FAQ retrieval capability of such a system. Further in this thesis, Section 2 deals primarily with the first whereas Section 3 with the second subject. Sections 4 and 5 discuss implementation and evaluation of both.

Section 2 describes the conceptual architecture and functionality of the EKD QA System. It starts with reasoning on various desired features of the system without going into implementation details. The reasoning is showed in the Goal Model of the system. The goals motivate the concepts, actors, and processes of the system.

Section 3 is devoted to the natural language processing and FAQ retrieval aspects of the EKD QA System. Namely, the Prioritized Keyword Matching technique has been developed in order to match a natural language question to FAQ entries in the database. The section describes the data concepts, informal and semi-formal algorithms, as well as the process of creating FAQ entries for Prioritized Keyword Matching. The notion of multiple lexicon, which brings the issue of context into the traditional understanding of a lexicon, is formulated.

Section 4 describes the implementation and operation of the EKD QA System. It starts with the implementation architecture of the system, which comprises 5 software components: the user interface tool, FAQ repository, query manager, transaction logs, and administration tool. Description of all these units follows. Requirements of the user interface and the maintenance of the FAQ repository are stated. Along with the data structure, the maintenance of the FAQ repository is discussed.

Section 5 presents evaluation of the Prioritized Keyword Matching technique, discusses completeness of the FAQ set and technical characteristics of the EKD QA System.

Section 6 gives an insight into possible directions of further research, and conclusion.

2 Conceptual Architecture and Functionality of the EKD QA System

This section discusses the main principles, desired features, concepts, and functionality of the EKD QA System without considering its implementation. EKD itself has been adapted and used to design and present the system's features and concepts. The notation, described in [EKD 1998], is considered easy to understand.

The following subjects are covered in this section:

- initial reasoning and motivation of the design of the system: what are the goals of the system, what features should it have, what are the problems and opportunities in the development of the system, how these issues interact with each other;
- conceptual architecture of the system;
- the basic processes and functionality of the system.

This section does not deal with the technical requirements of the system. The implementation of the system is discussed in Section 4. The nonfunctional requirements of the user interface are described in Section 4.2.2. The requirements to the administrator of the system are formulated in Section 4.3.3.

2.1 Objectives and Principles Driving the Design of the System

The idea of building the EKD QA System appeared at the Department of Computer and Systems Sciences at Stockholm University and the Royal Institute of Technology when the Elektra [ELEKTRA 1996] project started. The project assumed application of EKD in solving management problems in two power supply companies. The participants of the project came from academic and business environment of different countries: Greece, Israel, France, Sweden, and UK. In order to better exchange ideas about EKD, creating of an FAQ answering system that could answer questions asked in natural human language was suggested. Nonetheless, the proposal of such a system was not a part of the activities within the Elektra project. *The EKD QA System has no clearly defined customer*. The idea of building the system emerged in academic environment, and the biggest value of the system is its research outcomes – the approach, algorithms and data structures for automated FAQ answering. Hence, the requirements of the system are based on general reasoning about what the problems of FAQ answering are in general and what the EKD users (e.g., people using EKD in modelling their business, people involved in the development of EKD, students) could need in particular.

The conception of the system seemed interesting and challenging. The natural language interface would appeal to the users. Step by step, the system would

accumulate experiences of the use of EKD: the users formulate their questions expressing their problems and ask these questions to the system; the system collects them. The FAQ set operated by the system is based on these questions; FAQ answers give the solutions to the problems expressed in the FAQs. Obviously, the research value of the EKD QA System as an FAQ answering system is not EKD dependent.

The above reasoning leads to the principles of what kind of system we intend to build:

- *The system operates FAQs about EKD and Enterprise Modelling.*
- *The system operates its own independent set of FAQs.* Although Enterprise Modelling techniques are in use for years, questions and problem statements about them have not been gathered and published. Therefore new FAQs about EKD and Enterprise Modelling need to be created and maintained for the EKD QA System.
- *The system has a natural language based interface.* The system answers questions asked in ordinary English. This makes user navigation through the FAQ set easier. The questions are recorded, which lets the EKD developers find out what problems with EKD its users actually have.
- *The system is a Web-based application* because it has to be accessible from different locations all over the world. Besides, a Web-browser is a ready-to-use graphical user interface tool; there is no need to build another one.

Figure 2.1 on the next page presents refinements of these principles into the Goal Model of the EKD QA System. A “wish list” of goals shows the motivation of the main features of the system. The goals represented by boxes with a shadow are decomposed and discussed further. Goals, problems, constraints, and opportunities interact with each other. The prevailing relationship in a Goal Model is “supports”. This relationship is used to decompose goals and other components. “Goal 1 supports Goal 2” means that achieving of Goal 1 makes achieving of Goal 2 easier.

The abbreviation “NLP” stands for natural language processing.

Clarification of the components of the Goal Model follows.

The system is built in order to overcome the drawbacks of existing FAQ lists, (*Problems 1* to *5*; these problems were introduced in Section 1.1.2), which are the motivation of *Goals 1, 2* and *3* – the top-level objectives of the system. The EKD developers would like to find out what questions about EKD people have, to answer these questions, and to do that in a user-friendly way. These objectives are supported by *Goal 4* which is a formal statement that we want to build and maintain a computerized EKD question answering system. Goal 4 is supported by Goals 5, 6, and 7 which represent the main principles of the system.

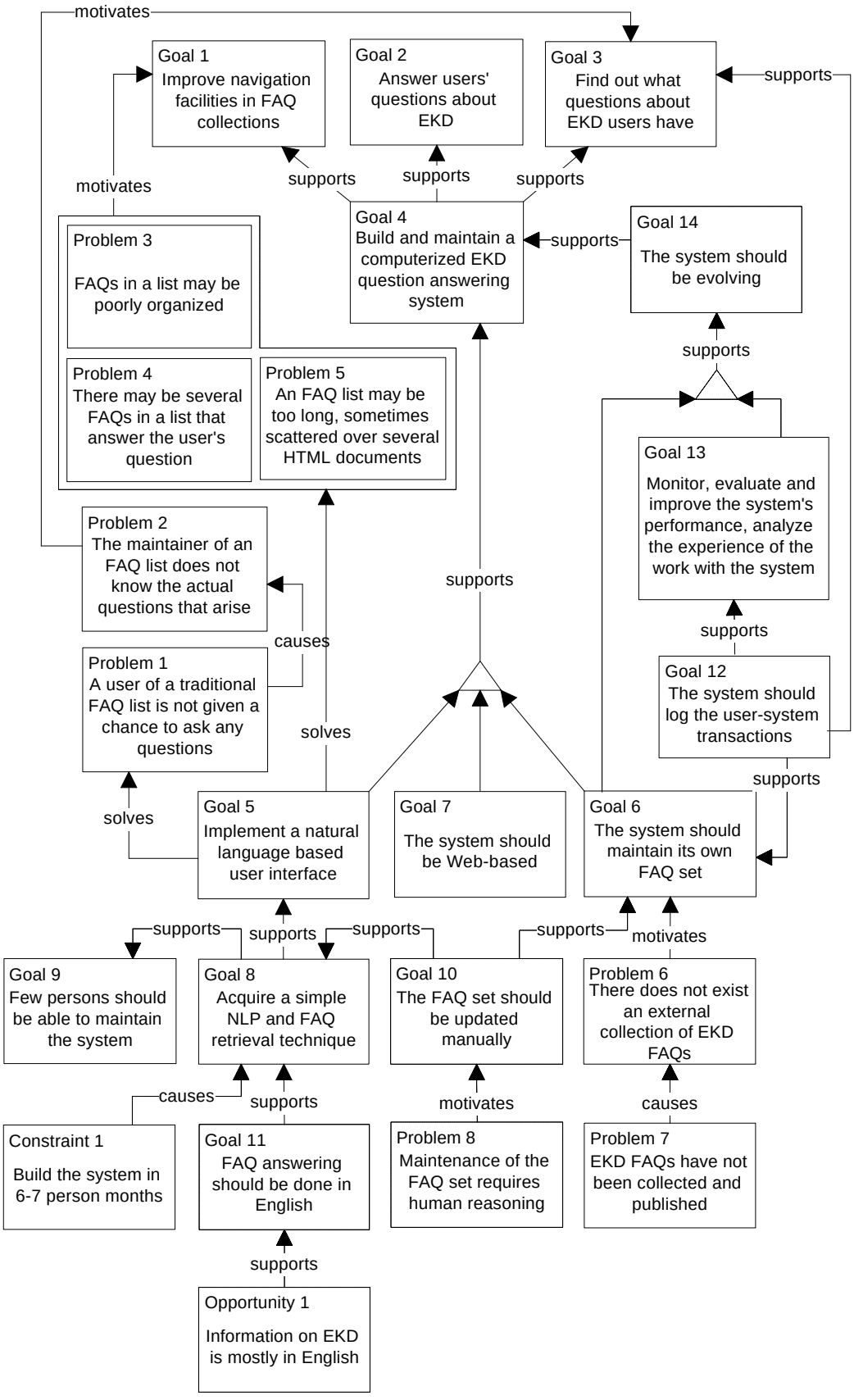

Figure 2.1 Objectives driving the design of the EKD QA System.

The system allows asking questions in natural language, which is implied in *Goal 5* (refined in Figure 2.2). In order to accomplish this task, we need to acquire an appropriate language processing technique (*Goal 8*, refined in Figure 2.4). The questions and answers are supposed to be in English (*Goal 11*) because it is an internationally used language. English is advantageous for computerized natural language processing because there has been a lot of related research done with English earlier. Another advantage is that words in English have fewer grammatical variations than, for instance, words in Russian and Latvian, and usually are not merged into compound words like in Swedish and German. *Opportunity 1*, which says that information on EKD is mostly in English, supports Goal 11. The language processing technique should be reasonably simple so that the system can be built and put into operation within few person months (*Constraint 1*), and maintained by few people (*Goal 9*). Constraint 1 is motivated by the need to describe the system in this thesis.

Goal 6 (refined in Figure 2.3) represents another important feature. Since the questions about EKD have not been formerly collected (*Problem 7*), there does not exist an external (i.e., outside the EKD QA System) source of EKD FAQs (*Problem 6*). Therefore the system maintains its own original set of FAQs. Formulation of the FAQs and finding their answers requires human reasoning (*Problem 8*); this research does not deal with automatization of this task. Therefore FAQ entries in the database should be created manually (*Goal 10*); computer assistance is available and welcome. The links between Goal 6 and Goals 12, 14 are explained further discussing the refinement of Goal 6. The link between Goals 10 and 8 is explained with the refinement of Goal 8.

The developers and users of EKD are located in different offices, cities, and even countries; the users of the system may be miles away from each other and from the system. Therefore, according to *Goal 7*, the system is built as a Web-based application. Advantages of the Web as a medium of communication were discussed in Section 1.1.1.

According to *Goal 12*, the system records the asked questions and other information concerning the interaction between the system and its users. This goal supports Goal 3 – the EKD developers would like to know what questions about EKD people have, and *Goal 13* – the developers of the system would like to monitor the system's performance (mainly the quality of FAQ retrieval) in order to evaluate and improve the system. The experience obtained while working with the system would be useful in order to build analogous systems.

The system is evolving (*Goal 14*). This is a consequence of Goal 6 – the system's ability to answer questions increases as more and more new FAQs are included into its database, and Goal 13 – the system's performance is supposed to improve.

Figure 2.2 on the next page shows refinement of Goal 5.

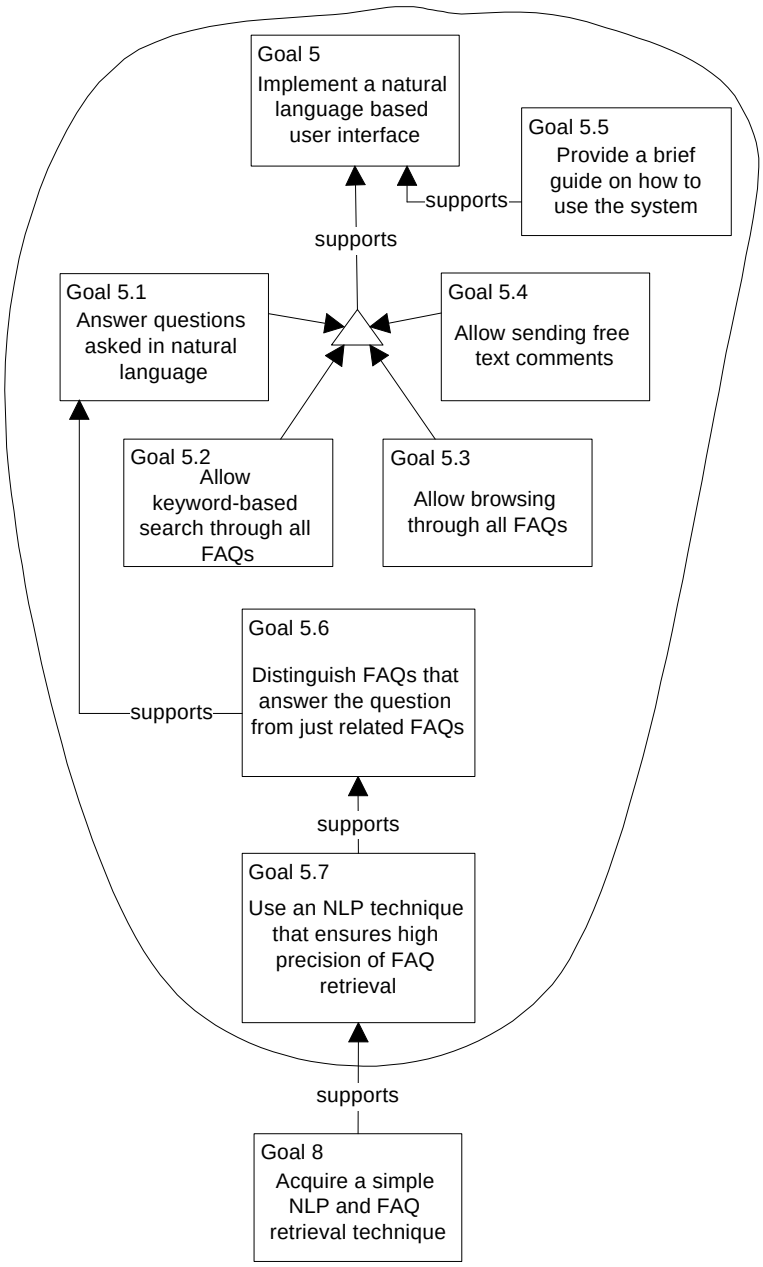

Figure 2.2 Refinement of Goal 5 "Implement a natural language based user interface".

The desired user interface functions are:

- *Goal 5.1*. A user can ask his or her question in ordinary English, the system replies by one or several corresponding FAQs and their answers, if any. This is the most desired and valuable user interface function, the reason why the development of the EKD QA System started. It aims at resolving the problems with existing FAQ lists, as stated in Section 1.1.2. Namely – to let people freely express their questions, to let information providers obtain these questions, and to improve (or at least make different) the navigation facilities through a collection of FAQs.

Traditional Information Retrieval systems respond to a user query by a list of documents of different relevance to the query, where the most relevant documents are in the top of the list. The EKD QA System distinguishes FAQs that answer the user's question from just related FAQs (*Goal 5.6*). Let us consider an example. A user asks the question "What is Business Rule modelling?" The system answers the question and suggests related FAQs:

  - A close answer: "What is the Business Rule Model?"
  - Related FAQs: "What is business rule?", "What is Enterprise Modelling?"

- *Goal 5.7*. The natural language processing and FAQ retrieval technique used by the EKD QA System should ensure so high quality of FAQ retrieval. Still, according to Goal 8 the technique should be reasonably simple.
- *Goal 5.2*. A user can perform keyword-based search in order to find FAQs in the database containing certain keywords.
- *Goal 5.3*. A user can browse through the list of all FAQs in the database. This feature does not deal with natural language processing.
- *Goal 5*.4. Users are welcome to send free text comments and recommendations concerning the system and EKD to the administrator of the system.

Goal 5.5 was introduced later after the initial system had been tested at work for a while. At the beginning, there was no user guide for the system since the use of the simple interface was supposed to be obvious. However, the users managed to make mistakes and the need for a brief "how to use it" guide became apparent.

Figure 2.3 shows refinement of Goal 6.

Problem 6 states that there does not exist any collection of FAQs about EKD outside the EKD QA System. Therefore the initial set of FAQs for the system has to be created from scratch (*Goal 6.1*) and continuously updated (*Goal 6.2*) as the "master copy" of the data. This is computer assisted manual work (Goal 10).

There are two main methods of finding the problem statements to be reflected in the FAQ set: scanning the environment of the EKD users and developers (*Goal 6.3*), which is specially important at the initial stage of populating the FAQ set, and checking the user-system transaction logs where all the users' questions are recorded (Goal 12). Maintenance of the FAQ set is discussed in Sections 4.3.3 to 4.3.6. Continuous updates of the FAQ set ensure that the system is evolving (Goal 14).

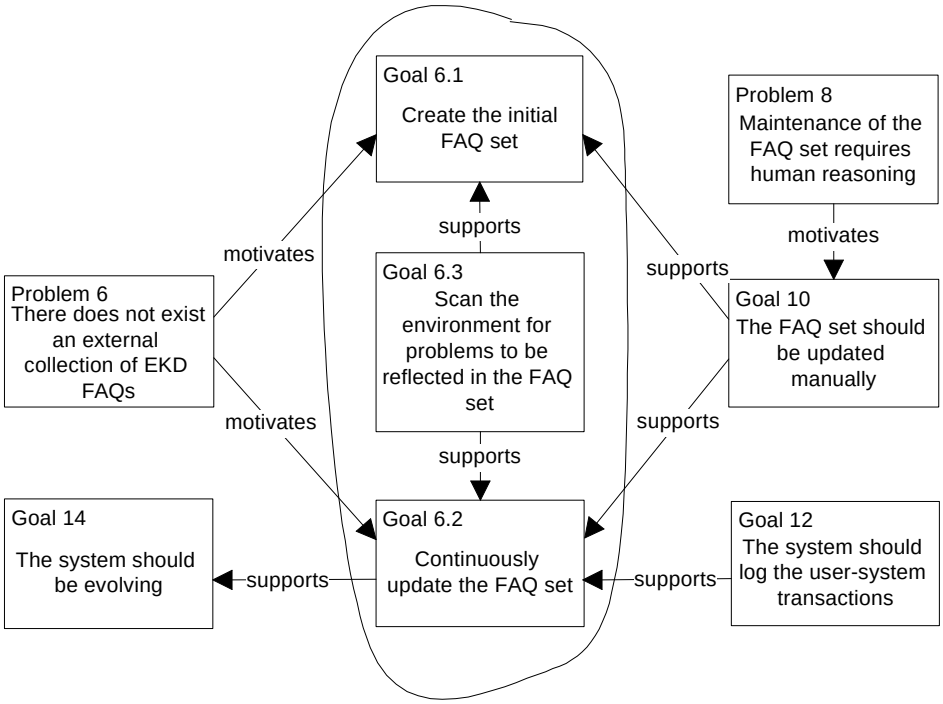

Figure 2.3 Refinement of Goal 6 “The system should maintain its own FAQ set”.

Figure 2.4 on the next page shows refinement of Goal 8.

According to a literature survey, most natural language question answering systems, such as FAQ Finder and the systems presented in [Grosz et al. 1986], deal with transformation of free text input into semantically well defined concepts that are further used to select appropriate information sources and to retrieve data. This requires syntactical and lexical analysis of user queries. The approach may yield high quality of FAQ retrieval if the analysis is done thoroughly. Thorough analysis requires a rich lexicon and a knowledge base dealing with the meanings of the FAQs. Both the lexicon and the knowledge base are complex and difficult to implement and maintain (*Problem 8.1*). FAQ Finder, for instance, does only limited lexical analysis of user queries in order to distinguish the concepts in these queries. Obviously, limited analysis cannot represent all the diversity of meanings of words and the relationships between concepts in different contexts. Therefore precision of such retrieval is too low if we want to distinguish a close answer from simply related FAQs (Goals 5.6 and 5.7 in Figure 2.2).

On the other hand, there exist statistical methods in Information Retrieval. These methods use term vectors where a vector shows frequency of the terms (in better case – concepts representing classes of synonyms) in the associated document. Similarity of two documents is calculated from the frequency of common terms in both documents. These methods were designed to compare large documents; single sentences are too short for calculations of term frequency (*Problem 8.2*). Another

difference of traditional Information Retrieval systems from the EKD QA System is that the latter does not have a goal to eliminate human participation in creating the index of documents (i.e., FAQs) because the formulation of the FAQs and their answers requires human reasoning. It is important to understand this difference in order not to misjudge the EKD QA System.

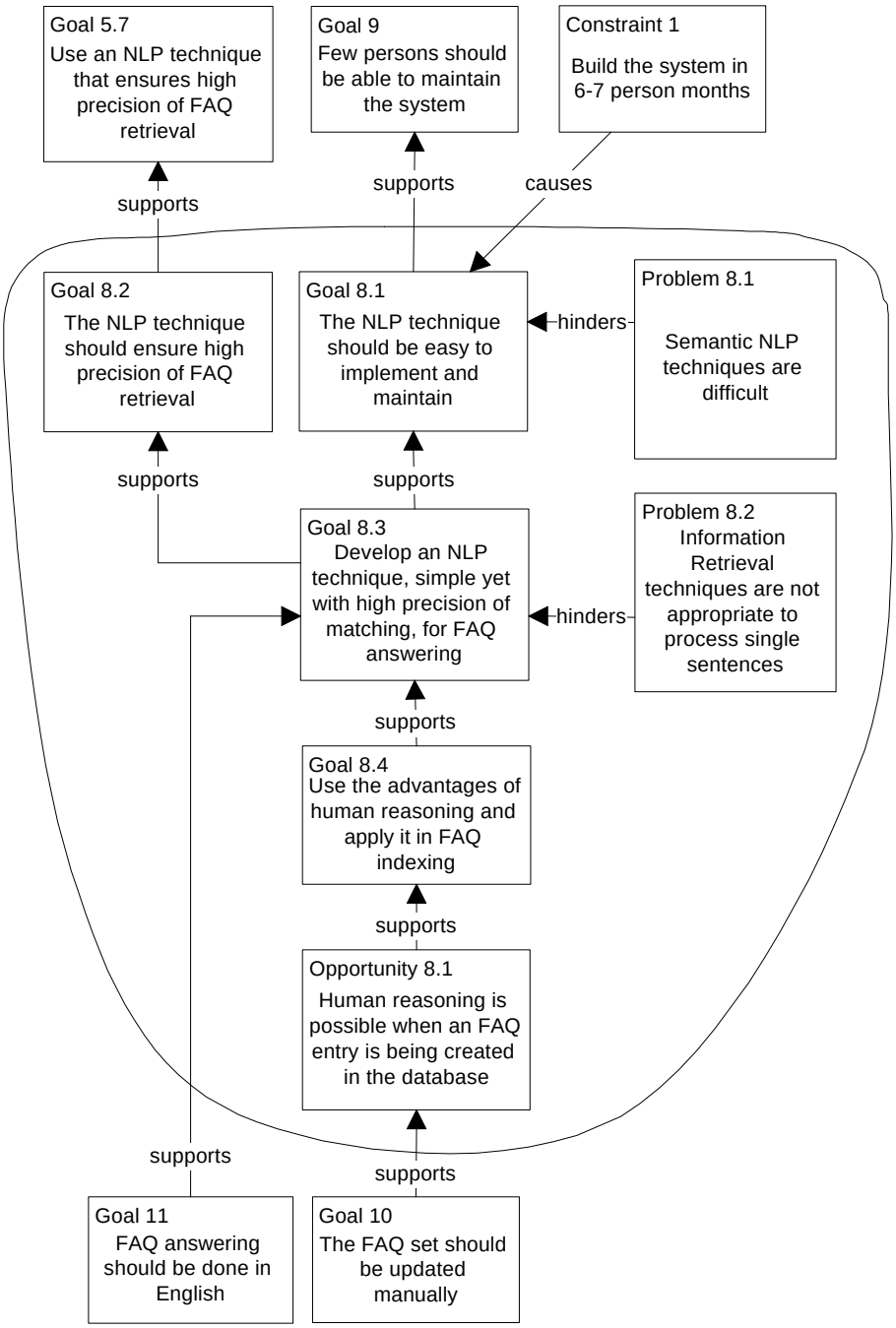

Figure 2.4 Refinement of Goal 8 “Acquire a simple NLP technique for FAQ answering”.

A new natural language processing and FAQ retrieval technique was developed within the scope of this research (*Goal 8.3*). This technique is called Prioritized Keyword Matching and is described in Section 3. Prioritized Keyword Matching has features from both Information Retrieval and semantic language analysis. Like in Information Retrieval, the technique proposes iteration through the database and matching a user's question to each FAQ by using the index of FAQs. Yet this index is more "intelligent" than term vectors: for each FAQ it contains representative keywords grouped into classes of context dependent synonyms; the classes present the concepts associated with the FAQ; the concepts are assigned different importance within the context of the FAQ. Human reasoning is used in order to create the index (*Opportunity 8.1* and *Goal 8.4*), which makes it possible to avoid a complex knowledge base and a big lexicon. The algorithms and data structures of Prioritized Keyword Matching are easy to implement and maintain (Goal 8.1), and has reasonably high quality of FAQ retrieval (Goal 8.2; Section 5.1 discusses evaluation of the FAQ retrieval).

2.2 Concepts, Actors and Roles Involved in the System

The EKD QA System is designed considering the traditional client-server architecture of Web-based applications where the user interface tool and the query processor are operated on separate computers and are connected through the Internet. The conceptual architecture of the EKD QA System (Figure 2.5 on the next page) identifies the key roles and concepts related to the system, and, therefore, characterizes the WWW environment around and within the system: who uses things, what things, what the system consumes and produces. The abbreviation "UI" stands for user interface.

Actor is an entity – thing or living creature – that performs some task. It is the original initiator of processes, producer of input, and consumer of output. There are four actors in the environment surrounding the system:

- *User* is a person who initiates all interactions with the system. The foremost objective of the system is to satisfy the needs of its users.
- *Web-browser* plays the role of the *user interface tool* which has two purposes:
  - to translate data from human-comprehensible form into machine-comprehensible form, and vice versa;
  - to send and receive data over the network.
- *HTTP server* is a gateway between the network and the main part of the system referred to as Actor 4. The system exchanges data with the network through the HTTP server.
- *System*, as referred to by Actor 4, is the main processor of user input and the database holder. Actor 4 interprets the input and replies to it.

It is difficult to draw a border where the actual system begins and where it ends. The Web-browser and HTTP server are also parts of the system as long as they implement the user interface and communication tools. Nonetheless, considering universal nature of WWW browsers and servers, they are categorized as separate actors.

- *Administrator* is a person whose main duty is to monitor the system's performance and update its database.

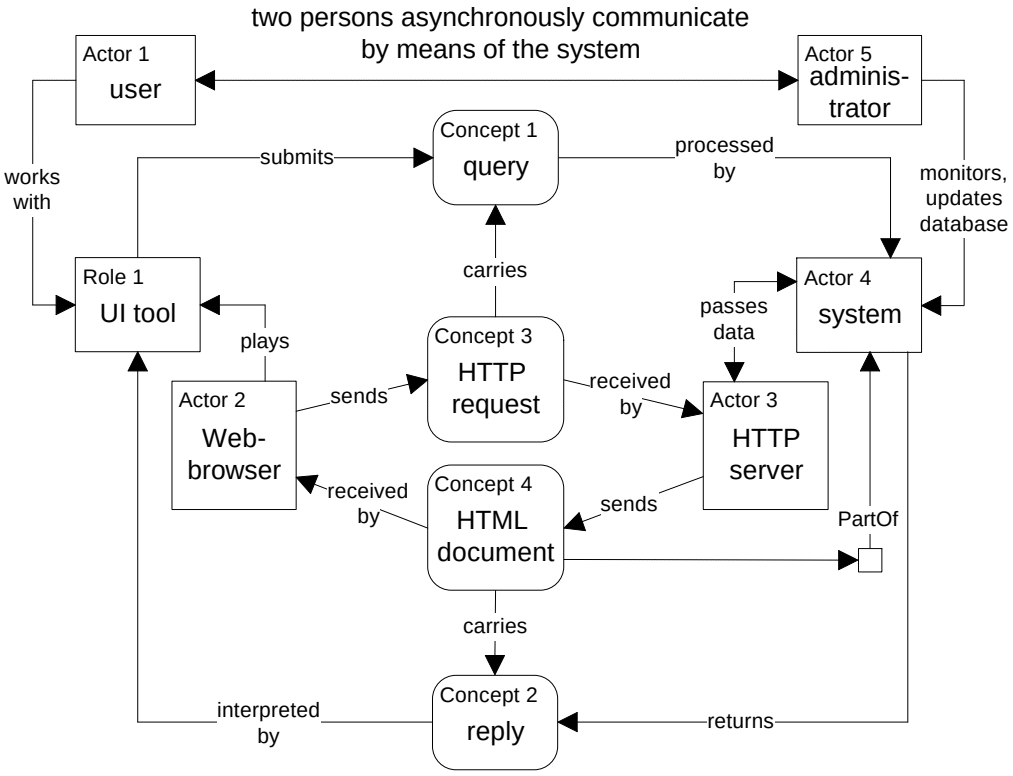

Figure 2.5 Environment with top level concepts, actors and roles surrounding the EKD QA System.

As we can see in the diagram, a user and the administrator asynchronously communicate with each other by using the system as a mediator. The conceptual architecture of the EKD QA System is a black box for the user; the user knows no more than his or her interface tool.

The concepts in Figure 2.5 embody the data flow between the system (Actor 4) and its user interface tool (Role 1):

- *Query* is the information that the user submits to the system: a question, keywords, comments, etc. The concept of query is refined in Figure 2.6.
- *Reply* is the system's response to the query. It may contain FAQs and their answers, error messages, etc. The concept of reply is refined in Figure 2.7.

- *HTTP request* is the carrier of the query encoded according to the protocol of communication between the Web-browser and HTTP server. The request may contain data from a fill-in form or just an order to send back an HTML document.
- *HTML document* is the carrier of the system's reply to the query. It can be a pre-stored document like the system's front page, or a document generated on the fly like the answer to a user's question.

  The HTML document is a part of the system because it contains:

  – HTML code of the user interface; and

  – a program script that performs limited validation of the user input before it is submitted to the system.

In Figure 2.5, we can observe two loops of relationships between the actors, role, and concepts:

- The outer loop "UI tool – query – system – reply – UI tool" deals with logical data representation in the communication between the user and the system: the user interface tool sends a query to the system that interprets it and returns a reply.
- The inner loop "Web-browser – HTTP request – HTTP server – HTML document – Web-browser" deals with physical representation of the data: the Web-browser sends an HTTP request to the server that sends an HTML document back to the browser after the information in the request is processed by the system.

The rest of this section deals with refinement of the basic data structures in the environment around the EKD QA System.

Figure 2.6 shows refinement of the concept of query.

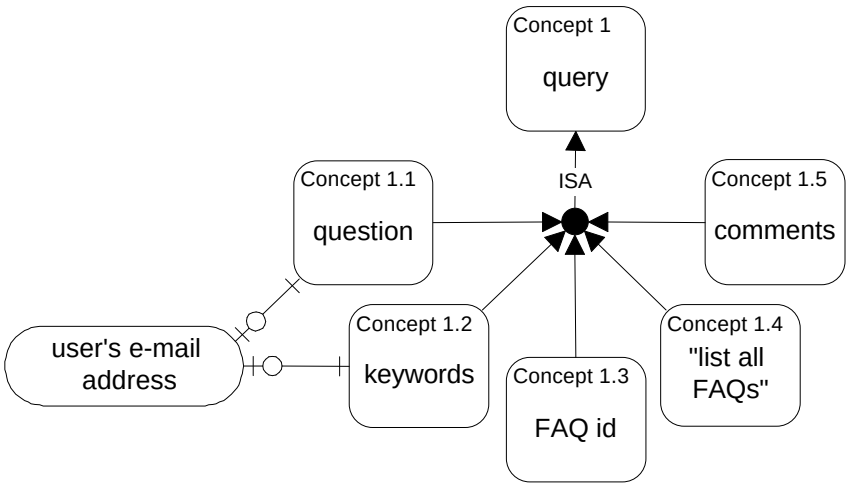

Figure 2.6 Refinement of Concept 1 "query".

Five types of queries are derived from the refinement of Goal 5 “Allow natural language user interface” in Figure 2.2. A user may submit:

- *question* stated in ordinary English in order to get one or several corresponding FAQs and their answers;
- *keywords* in order to retrieve FAQs containing certain words;
- an *FAQ identifier* in order to retrieve a specific FAQ and its answer;
- a request to *list all the FAQs* in the database;
- free text *comments*.

The dialog between a user and the system is logged (Goal 12). Therefore the user is recommended (but not required) to submit his or her e-mail address along with the question or keywords. This would let the administrator of the system contact the user if necessary. When sending free text comments, a user identifies him- or herself in the free text input.

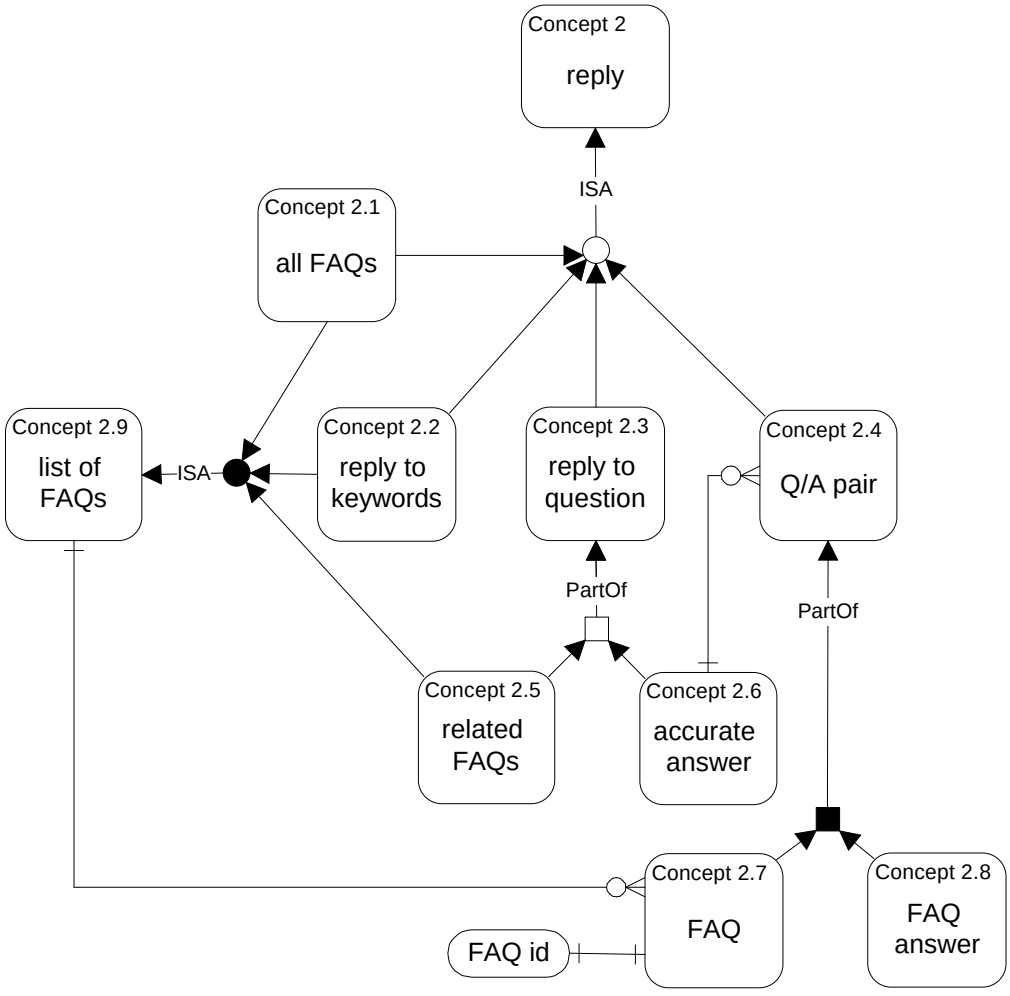

Figure 2.7 Refinement of Concept 2 “reply”.

Figure 2.7 shows refinement of the concept of reply. Each type of query requires its own type of reply:

- The simplest reply is a list of *all the FAQs* in the database.
- *Reply to keywords* is a list of FAQs matching the keywords.
- *Reply to a question* contains:
  - an *accurate answer*, if any, which is one or several *question / answer pairs* that answer the user's question, where each pair is an *FAQ* and *its answers*,
  - a list of *related FAQs*, if any.
- An individual *question / answer pair* is retrieved from a list using the FAQ identifier.

List of FAQs is a form of presentation of FAQs. The list consists of a sequence of FAQs where each FAQ has a unique identifier.

The correspondence between the queries and replies is showed in a process diagram in Figure 2.10.

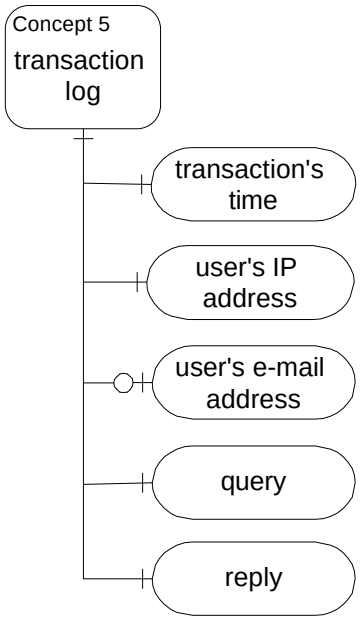

Figure 2.8
Transaction log.

According to Goal 12, the dialog between a user and the system is recorded in order to find out the users' needs and to monitor the system's performance. Figure 2.8 shows the structure of such a record. It contains the following information:

- The *time of the transaction* is recorded in order:
  - to see when the user has used the system; and
  - to make analysis of a sequence of queries from the same user.

  A chain of questions from the same person may be meaningful and interesting. The way of reasoning may show what people do and do not understand in the FAQs provided.
- *IP address* identifies a user even if he or she has not given the e-mail address. The main reason to identify a user is that described just above – to see a chain of queries coming from the same user.
- The reason to record the user's *e-mail address* was already explained when we discussed the structure of a query.

- All *queries* are recorded in order to see the performed actions. Especially interesting are questions. One of the top goals of the system is to find out what questions about EKD people have. If questions are not answered, they identify "holes" in the content of the database; in order to update the database by new FAQs, the administrator of the system must know the asked questions.
- The system's *reply* is logged in order to establish the correspondence between it and the submitted question or keywords. This is necessary for evaluation of the system.

2.3 Processes and Functionality of the System

2.3.1 Processes

The top-level processes showed in Figure 2.9 on the next page are derived from the conceptual architecture of the EKD QA System in Figure 2.5. The system is idle until the user interface tool sends a query to Process 1 which processes the query, sends back the reply, and passes a transaction log to be recorded. From now, the system is idle again.

Figure 2.10 shows refinement of Process 1. Processes 1.1 to 1.5, as well as Process 3 identify the functionality of the system.

2.3.2 Functionality

The following description of the functionality of the EKD QA System has references to Concepts 1 and 2 refined in Figures 2.6 and 2.7 respectively.

User Interface

1. *Question answering* (Process 1.4). A user submits a question stated in natural language (Concept 1.1); the system searches through its database in order to find one or several relevant FAQs and replies to the question (Concept 2.3).

   For each FAQ in the database:

   - if the question and FAQ are semantically very close, the system returns both the FAQ and its answer in the accurate answer to the question;
   - if the question and FAQ are not semantically very close but are still related, the system returns the FAQ and its identifier in a list of related FAQs.

   If the question has no accurate answer, the system returns an appropriate message.

   The notion of "question and FAQ are semantically close" is defined by the natural language processing technique discussed in Section 3.

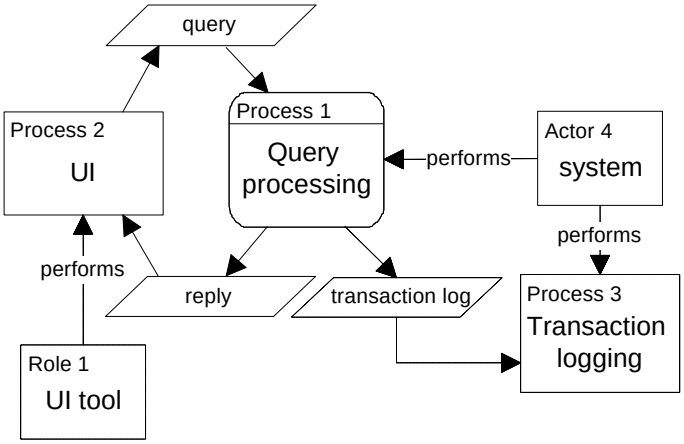

Figure 2.9 Top level processes of the EKD QA System.

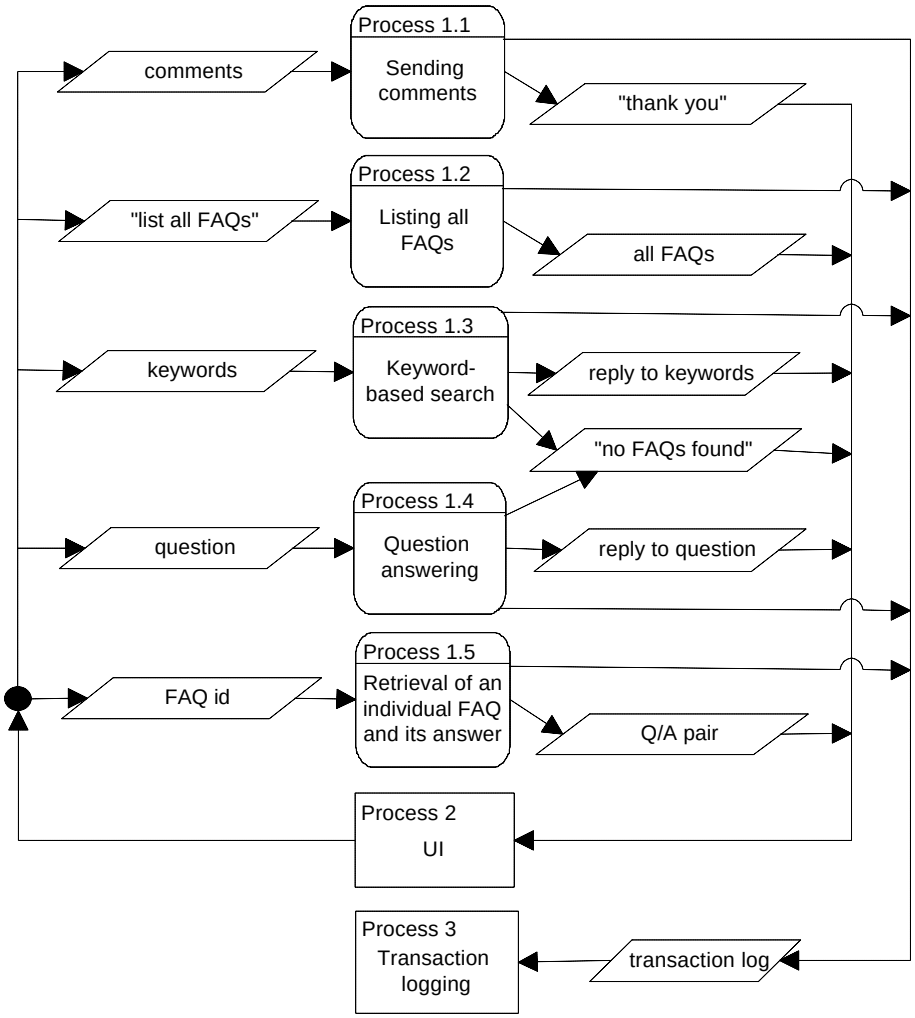

Figure 2.10 Refinement of Process 1 “Query processing”.

2. *Keyword-based search* (Process 1.3). A user submits one or several keywords (Concept 1.2); the system searches through its database in order to find FAQs that match these keywords. For each matching FAQ, the system returns the FAQ and its identifier in a list (Concept 2.2). If there are no matching FAQs, the system returns an appropriate message.

   A user selects one of two matching patterns:

   - $keyword_1$ is represented AND $keyword_2$ is represented AND … $keyword_n$ is represented; or
   - $keyword_1$ is represented OR $keyword_2$ is represented OR … $keyword_n$ is represented.

   The notion of "$keyword_i$ is represented" is implementation dependent and explained in Section 4.4.2.

3. *Listing all FAQs* (Process 1.2). Upon a user's request (Concept 1.4) the system lists all the FAQs in the database. For each FAQ, the system returns the FAQ and its identifier in a list (Concept 2.1).

4. *Retrieval of an individual FAQ and its answer* (Process 1.5). Having an FAQ and its identifier (typically in a list of FAQs), a user submits the identifier (Concept 1.3). The system returns an individual FAQ (the same that the user already has) and its answer (Concept 2.4).

5. *Sending comments* (Process 1.1). A user submits free text comments (Concept 1.5).

Transaction Logs

6. *A dialog between a user and the system is logged* (Process 3). The system records a transaction log for every user interaction with the system (Concept 5 in Figure 2.8).

3 Natural Language Processing and FAQ Retrieval Technique of the EKD QA System

The most important function of the EKD QA System – question answering – requires an appropriate natural language processing and FAQ retrieval technique. A brief insight into the question answering was already given in Section 1.3:

1. A user asks his or her question using an input form showed in Figure 1.2 and submits the question to the system.
2. After the system receives the question, it searches through its database in order to find one or several FAQs that are semantically close to the question. For each FAQ in the database, the system *matches the question to the FAQ in the database using the Prioritized Keyword Matching technique* (to be introduced), as showed in Figure 3.1.
3. After one or several, if any, relevant FAQs are found, the system returns them and their answers to the user, as showed in Figure 1.3. Hopefully, while reading the returned FAQs and their answers, the user will find the answer to his or her original question.

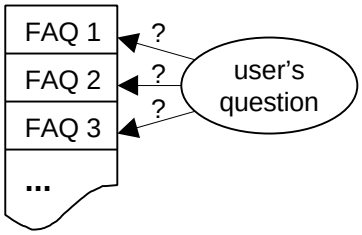

Figure 3.1
Application of natural language processing in the EKD QA System: is the question similar to an FAQ?

The system does not need to comprehend an arbitrarily asked question in order to formally match it to FAQs in the database. Therefore the Prioritized Keyword Matching technique uses shallow language understanding (see Section 1.4 for the explanation of the notion of shallow language understanding). The rest of this section presents Prioritized Keyword Matching, namely:

- the conceptual data structures of an FAQ entry in the database and a users' question;
- the algorithm used to match a particular user's question to a particular FAQ in the database;
- how to create new FAQ entries in the database;
- advantages and disadvantages of the proposed natural language processing technique.

3.1 Prioritized Keyword Matching

The task of the Prioritized Keyword Matching technique in the EKD QA System is to determine semantic similarity between *one* user's question and *one* FAQ in the

database.

3.1.1 Basic Idea

The idea of Prioritized Keyword Matching is based on the assumption that there are three types of words in a sentence within a certain context in a certain subject:

- *Primary keywords* are the words that convey the essence of the sentence. They cannot be ignored.
- *Secondary keywords* help to convey the meaning of the sentence but can be omitted without changing the essence of the sentence. The nuances may change though.
- *"Irrelevant" words*, like "a", "the", "is", etc., are words that are too common in ordinary language or the subject. The meaning of "irrelevant" words is close to that of stop-words in Information Retrieval. The only difference is that stop-words are always unimportant in a given collection of documents, whereas "irrelevant" words, although usually unimportant, may suddenly become relevant for a particular sentence in a given collection of sentences.

Let us consider an example with "What is the relationship between Business Goal Models and Business Process Models?" In this sentence, we distinguish:

- primary keywords "relationship", "goal", "process";
- secondary keywords "business", "models";
- irrelevant words "what", "is", "the", "between", "and".

If we modify this selection of words with their *synonyms and various grammatical forms*, we obtain a new, broader selection of words, which characterizes a set of different sentences that are semantically related to the one given above. We assume that these sentences are related although we do not comprehend them.

Let us define that each keyword is always represented by a number of synonyms and their grammatical forms, and that irrelevant words are the same for all the sentences. Hereby, if the same primary and secondary keywords can characterize two sentences, we declare that both sentences have about the same meaning, i.e., they match each other.

This is the basic idea of Prioritized Keyword Matching.

One may wonder why the words "business" and "models" in the above example are secondary keywords, i.e., less relevant. The reason is that Business Goal Models and Business Process Models are often referred to as simply goals and processes. A user may formulate the question as follows: "What is the relationship between goals and processes?" "Business" and "models" do not appear in this formulation.

The current implementation of the technique in the EKD QA System does not process phrases.

3.1.2 Data Concepts

Let us assume that we have a database consisting of FAQ entries where each FAQ has its primary and secondary keywords identified (Figure 3.2).

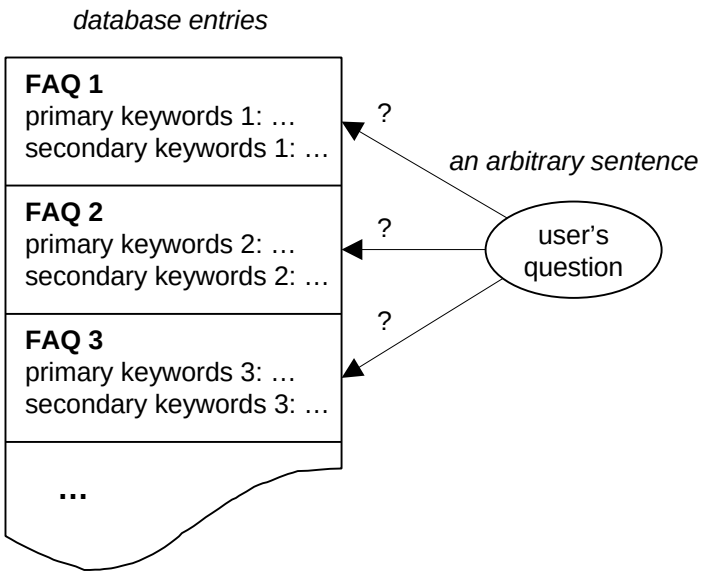

Figure 3.2 FAQs with identified primary and secondary keywords vs. an arbitrary user's question.

According to the basic idea of Prioritized Keyword Matching, an FAQ becomes a pattern that identifies a class of questions with similar meanings, where the primary and secondary keywords of the FAQ identify the concepts relevant to this pattern. After an arbitrary user's question is asked, we use the Prioritized Keyword Matching algorithm to match the question to each FAQ entry separately in order to determine whether or not the question belongs to the class of questions identified by the FAQ. Hereby, the algorithm has the following input:

- an arbitrary sentence (a user's question); and
- an FAQ entry with primary and secondary keywords.

The output of the algorithm is a statement denoting whether or not the user's question matches the FAQ in the entry. The algorithm uses also a list of "irrelevant" words introduced earlier; there is one list for all the FAQ entries in the database.

Figure 3.3 on the next page shows the concepts involved in Prioritized Keyword Matching and the relationships between these concepts if the user's question matches the FAQ as a *close answer*. This is not a picture of the data structure as implemented in the software.

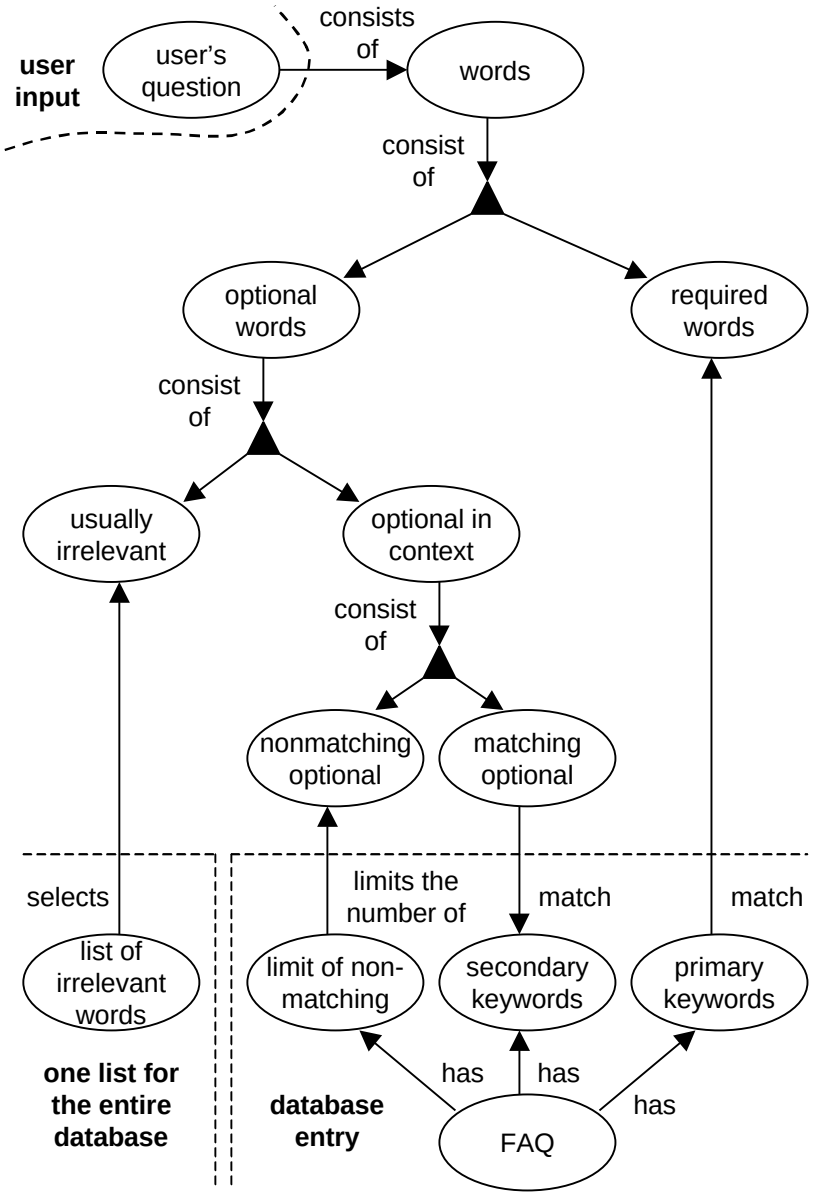

Figure 3.3 Concepts involved in Prioritized Keyword Matching when the FAQ *answers* the user's question.

It is important to note that the **only user's concern is his or her own question**. When typing the question, the user knows nothing about the structure of the database, required and optional words, primary and secondary keywords, the matching algorithm. All the data, except the question itself, either come from the database or are created during the matching process.

The *user's question* consists of *words*. Some of these words are *required* and match the primary keywords of the FAQ. Other words are *optional*. Among optional words, there are words that are *usually irrelevant* for all the FAQs in the database (e.g., "a", "is", "it", etc.). Others are *optional only within the context* of this

particular FAQ. Most of them match the secondary keywords of the FAQ. If, however, there are a few optional words that do not match the secondary keywords, their number is restricted by the *limit of nonmatching* words, which is specific for each FAQ.

One may wonder in which cases a usually irrelevant word may become relevant. Let us consider an example. If we want to distinguish two imaginary FAQs "What is EKD?" and "What was EKD?", then the usually unimportant words "is" and "was" become relevant and must be included as primary keywords in both FAQ entries respectively.

3.1.3 Informal Description of the Algorithm

In order to better understand Prioritized Keyword Matching, let us observe the algorithm together with an example. A user has asked a question; the system matches this question to an FAQ entry in the database.

User's question:

"How are substantial business goals related to business processes?"

FAQ:

"What is the relationship between Business Goal and Business Process Models?"

- *Primary keywords*:
  1. "goal", "goals";
  2. "process", "processes";
  3. "relation", "relations", "relationship", "relationships", "dependence", "dependencies", "connection", "connections", "association", "associations", "link", "links", "linked", "linking", "relate", "relates", "related", "relating", "connect", "connects", "connected", "connecting", "associate", "associates", "associated", "associating".
- *Secondary keywords*: "business", "businesses", "model", "models".
- *Limit of nonmatching words*: 1 (see the description in Step 5 of the algorithm).

The human common sense says that the user's question and FAQ convey roughly the same meaning. The system has to determine this by performing the following steps:

1. The system splits the user's question into separate words, ignoring repeated words.

   In the example, the user's question is split into:

- "how", "are", "substantial", "business", "goals", "related", "to", "processes".

2. The system matches the ***primary keywords*** in the entry to the words of the user's question. Primary keywords, usually two or three, express the essence of the FAQ in the entry and are expected to be fully represented in the user's question. If there is at least one primary keyword that is *not* represented among the words of the user's question by at least one synonym or grammatical form, the system *rejects* the match between the user's question and the FAQ.

   In the example, all three primary keywords of the FAQ are represented among the words of the user's question: "goals" (1), "processes" (2), and "related" (3). Therefore there is no reason to reject the match between the user's question and the FAQ at this stage. The system takes away the required words and proceeds with the optional words:

   - "how", "are", "substantial", "business", "to".

3. From the optional words, the system filters out ones listed as usually irrelevant ("a", "the", "is", etc.). The filtering is based on the ***list of irrelevant words***. The system has one such list for all the FAQs in the database. If the same word is named both in the list of irrelevant words and among primary keywords, this word has already been identified in the previous step and is not important anymore.

   In the example, irrelevant are the words "how", "are", "to". After they are filtered out, there are only context dependent optional words left:

   - "substantial", "business".

4. The system matches the context dependent optional words of the user's question to the ***secondary keywords*** in the entry. Secondary keywords convey nuances of the meaning of the FAQ. Unlike the words named in the list of irrelevant words, secondary keywords are optional in representation of a particular FAQ; in representation of some other FAQ the same words may be required or not welcome at all. If some context dependent optional words match the secondary keywords, the system identifies these words and filters them out.

   In the example, the secondary keywords are "business", "businesses", "model" and "models". There is one context dependent optional word represented among the secondary keywords – "business". The other one is not:

   - "substantial".

5. The system considers the word or several words left – those words of the user's question that do not match primary nor secondary keywords, and are not in the list of irrelevant words. If there are too many such words, the system *rejects* the match between the user's question and the FAQ in the entry. How does the

system determine this “too many”? For this purpose, there exists a ***limit of nonmatching words*** (in practice 0 or 1, formally any number) stated in the entry. The number of nonmatching optional words of the user’s question may not exceed this limit.

In the example, the only nonmatching optional word is “substantial”, which does not exceed the limit of nonmatching words in this FAQ entry set to 1. Therefore there is no reason to reject the match between the user’s question and the FAQ.

6. Already twice the system had an opportunity to reject the match – in Steps 2 and 5. It did not use this opportunity. It *accepts* the match between the user’s question and the FAQ in the entry.

There are two judges for each FAQ entry in the database that determine whether or not the user’s question matches the FAQ in the entry:

- *Primary keywords* in the entry determine which words must be present in the user’s question.

  “What is the difference between the business goal and process models?” does not match the FAQ in the example above because the primary keyword (3) is not represented.

- *Limit of nonmatching words* restricts the number of those words of the user’s question that do not match primary nor secondary keywords in the FAQ entry, and are not listed as usually irrelevant words either.

  “How are business goals related to the processes in French cooking books?” does not match the FAQ in the example above because there are too many nonmatching words: “French”, “cooking”, “books”.

As an apparent simplification of language processing, the algorithm does not process a phrase as a semantically unified sequence of words. Such ignorance is fine in most cases. Nonetheless, the algorithm cannot distinguish, for instance, “process modelling” from “modelling process”. It is planned to process phrases in future work.

Related FAQs

A user would loose much information if the system retrieved only FAQs that are close answers to his or her question. Therefore the system retrieves related FAQs as well. Figure 3.4 shows the relationships between the concepts already introduced in Figure 3.3 if the FAQ matches the user’s question as a *related FAQ*, not as a close answer. An FAQ is considered related to the user’s question if all its primary keywords are represented among the words of the user’s question. This is checked in Steps 1 and 2 of the above algorithm.

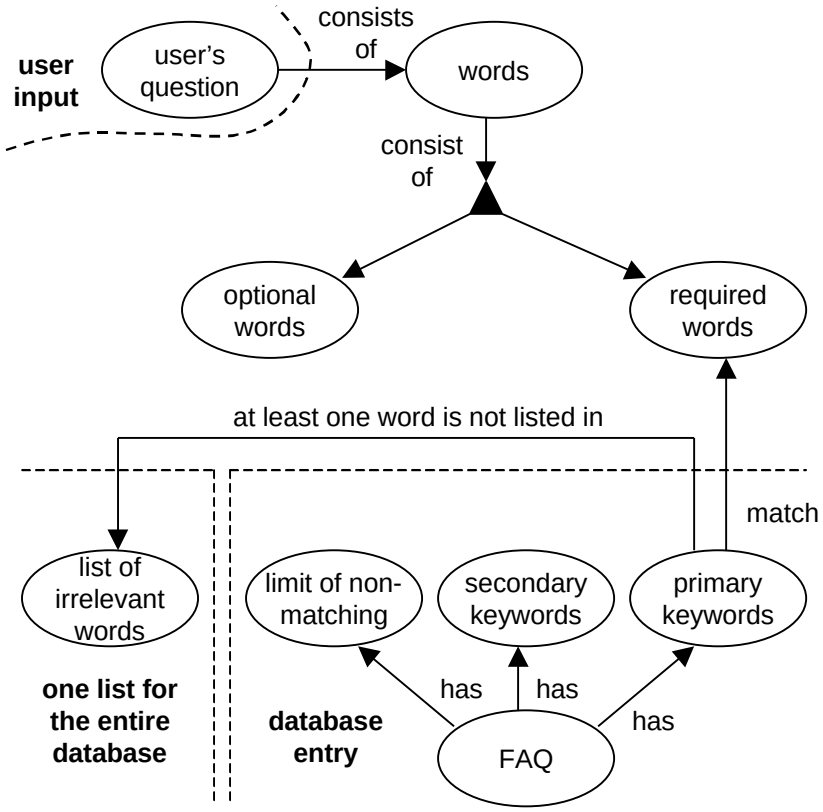

Figure 3.4 Concepts involved in Prioritized Keyword Matching when the FAQ is *simply related* to the user's question.

After this definition of related FAQs was implemented in the EKD QA System, almost every question got "What is EKD?" as a related FAQ, which became annoying. In order to avoid too general related FAQs, the system makes sure that at least one primary keyword in the entry is *not* named in the list of irrelevant words before suggesting the FAQ as related to the user's question. Since "EKD" was in this list, it helped.

3.1.4 Semi-Formal Algorithm

This section presents a semi-formal version of the Prioritized Keyword Matching algorithm, which can be considered for implementation in software. The algorithm is written in a language based on Pascal modified in order to improve readability rather than efficiency of an imaginary executable code.

There are remarkable differences between the formal and informal versions of the algorithm. The objective of the informal version (steps 1 to 6 in the previous section) along with the data concepts (Figure 3.3) is to illustrate the principles of Prioritized Keyword Matching, whereas the semi-formal version focuses on performance of the actual matching. Many data concepts introduced in Figure 3.3 do not appear in the semi-formal version of the algorithm.

First, let us consider the data in an FAQ entry needed for the matching algorithm.

```
Tentry = record
   Primary: array of (array of string);
   Secondary: array of string;
   Limit: integer;
end;
```

The data representing an FAQ are `Primary` – array of primary keywords, each is an array of synonyms and grammatical forms, `Secondary` – array of secondary keywords along with synonyms and grammatical forms, and `Limit` – limit of the number of nonmatching words. The matching algorithm does not use the text of an FAQ. One may notice that the data type of primary keywords (array of array of string) differs from that of secondary keywords (array of string). For primary keywords, a grouping of synonyms and grammatical forms of each separate keyword is necessary, whereas for secondary keywords such a grouping is not needed since there is no requirement for every secondary keyword to be represented in the user's question.

The main routine of the algorithm, called `Match`, takes an arbitrary question and an FAQ entry described above and returns "true" (if the question and the FAQ match) or "false" (if they do not match).

```
function Match(Question: string; Entry: TEntry): boolean;
var
   Words: array of string; // words of the user's question
   Nonmatching: integer;   // number of nonmatching
                           // relevant words
   i: integer;
begin
   Words := SplitIntoWords(Question);
   if not IsEachPrimaryRepresented(Words, Entry.Primary)
      then return false;
   // count nonmatching words
   Nonmatching := 0;
   for i := 1 to length(Words) do begin
      if not ( IsIrrelevant(Words[i]) or
               IsAmongPrimary(Words[i], Entry.Primary) or
               IsAmongSecondary(Words[i], Entry.Secondary)
             ) then Nonmatching := Nonmatching + 1;
   end;
   if Nonmatching > Entry.Limit then return false;
   return true;
end;
```

The subroutine `IsEachPrimaryRepresented` checks whether or not each primary keyword is represented among the words of the user's question by at least one synonym or grammatical form.

```
function IsEachPrimaryRepresented(Words: array of string;
                     Primary: array of (array of string)
                     ): boolean;
var
   Synonyms: array of string;
   i, j, k: integer;
begin
   for i := 1 to length(Primary) do begin
      Synonyms := Primary[i];
      for j := 1 to length(Synonyms) do begin
         for k := 1 to length(Words) do begin
            if Synonyms[j] = Words[k] then goto #Next;
         end;
      end;
      return false;
   #Next:
   end;
   return true;
end;
```

The following are supporting subroutines.

```
function SplitIntoWords(Text: string): array of string;
begin
  // the function takes a piece of text and splits it into
  // an array of separate words; returns this array
  …
end;

function IsIrrelevant(Word: string): boolean;
begin
  // the system checks whether the parameter Word is or is
  // not represented in the system's list of generally
  // irrelevant words;
  // returns true or false accordingly
  …
end;
```

```
function IsAmongPrimary(Word: string;
                        Primary: array of (array of string)
                        ): boolean;
var
   Synonyms: array of string; // one primary keyword
   i, j: integer;
begin
   for i := 1 to length(Primary) do begin
      Synonyms := Primary[i];
      for j := 1 to length(Synonyms) do begin
         if Synonyms[i] = Word then return true;
      end;
   end;
   return false;
end;

function IsAmongSecondary(Word: string;
                          Secondary: array of string
                         ): boolean;
var
   i: integer;
begin
   for i := 1 to length(Secondary) do begin
      if Secondary[i] = Word then return true;
   end;
   return false;
end;
```

3.2 Creating FAQ Entries

While discussing the construction of FAQ entries, we will prescind from the quality of information in FAQs and their answers. Our concern now is: if a user has asked a question, and there is a corresponding FAQ in the database, how must the FAQ entry be organized in order to match as many different forms of the user's question as possible. The Prioritized Keyword Matching technique is used.

According to Goal 6 in Figure 2.1, the EKD QA System maintains its own set of FAQs rather than an index of some external collection. According to Problem 8 and Goal 10, creating and maintenance of the FAQs in the database requires human reasoning and is a manual computer assisted process. This section explains what data are needed in an FAQ entry in order to ensure high quality of automated retrieval of the created entries.

What is a good FAQ entry? There is a simple answer: a good FAQ entry is one which *does* match a large variety of differently formulated users' questions with about the same meaning, and *does not* match not related users' questions. Three features characterize a good entry:

- *Sufficient number of auxiliary entries* helps to meet a large number of possible formulations of a user's question.
- *Good selection of primary and secondary keywords* in an entry highlights representative concepts of the FAQ.
- *Good controlled vocabulary – selection of synonyms and grammatical forms –* of the keywords increases flexibility of the keywords.

The process of selecting auxiliary entries, the keywords, and their controlled vocabulary corresponds to the process of creating term vectors in Information Retrieval and is called indexing. There may be a number of methods of how to create FAQ entries for Prioritized Keyword Matching; this research does *not* focus on optimization of this process.

An example of an FAQ and its keywords was already presented in Section 3.1.3 along with the informal description of the matching algorithm. The maintenance of the FAQ set of the EKD QA System is discussed in Sections 4.3.3 to 4.3.6.

3.2.1 Auxiliary Entries

One can formulate the same question differently. Although the approach with primary and secondary keywords is rather flexible, sometimes one FAQ entry in the database cannot represent all possible formulations of the corresponding users' questions. Therefore the administrator of the system begins with the following two steps of creating the entry:

1. Express the given FAQ in different ways in order to meet various user formulations. A user may ask weird questions. The system should correctly recognize those that sound natural, are concise, and have a clearly stated issue.
2. Group the formulations so that the words there differ only by synonyms and grammatical forms but the structure of the sentences remains the same. A group represents an auxiliary entry; the same primary and secondary keywords can characterize all the sentences in the group.

Several auxiliary entries in the database may represent one FAQ; often only one entry is needed for an FAQ. Each auxiliary entry operates its own keyword selection which is:

- broad enough to match as many different user formulations of the questions as possible (then we need fewer auxiliary entries); and
- narrow enough to match as few unrelated questions as possible.

Figure 3.5 shows an example of three questions having the same answer. These questions may need separate auxiliary entries in the database.

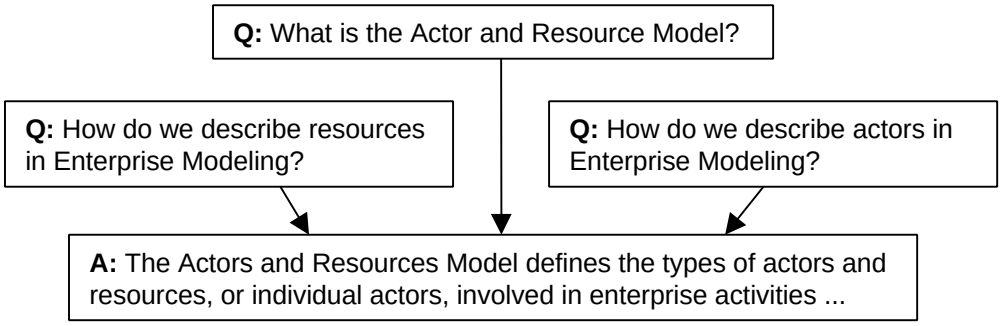

Figure 3.5 Three questions having the same answer.

We may try to put all three questions into one entry and provide this entry with a broad selection of keywords. However, a broad selection of keywords has a poor ability to distinguish similar but not related users' questions. Therefore several auxiliary entries with narrower selection of keywords are preferred.

Certain mental exertion based on one's imagination, experience and common sense is needed in order to find and group different formulations of an arbitrary FAQ. Unfortunately, easy computerization of this task is not expected in the nearest future.

In the EKD QA System, an entry is implemented as a single record in a single database table.

3.2.2 Selecting Keywords

After the auxiliary entries of a given FAQ are identified (Steps 1 and 2 in the previous section), the administrator takes the third step:

3. For each auxiliary entry, select the keywords that best represent the formulation of the FAQ in this entry:
   a) select the primary keywords;
   b) filter out "irrelevant" words;
   c) select the secondary keywords;
   d) set the limit of nonmatching words, usually 0 or 1, which depends on the length and complexity of the formulation of the FAQ.

Synonyms and various grammatical forms of the keywords are considered so that the entry covers as many different ways of asking the same question as possible. In case of the EKD QA System, the administrator selects these words manually and using the administration tool (see Section 4.6) enters them directly into the database record along with other data (the text of the FAQ itself, its answer, other implementation dependent information).

Grammatical Forms

While examining the grammatical diversity of a word, the administrator of the system is recommended to consider the following cases:

- singular and plural forms of nouns: “model”, “models”;
- tenses of verbs: “do”, “does”, “did”, “done”, “doing”;
- different spelling of words, American English vs. British English: “mode**l**ing” vs. “mode**ll**ing”, “analy**z**e” vs. “analy**s**e”, “formula**s**” vs. “formula**e”**;
- split or merged words: “sub-model” vs. “submodel”, “non-existent” vs. “nonexistent” vs. “not existent”.

Usually, there are not so many adjectives and adverbs in user asked questions. Many of them are categorized as irrelevant words; others should be used as secondary keywords. All pronouns are categorized as irrelevant words.

Context Dependent Synonyms

Four cases of synonymy should be considered when a keyword is analyzed:

- ordinary language synonyms: “related”, “connected”, etc.;
- switching between related verbs, nouns and adjectives: “In what cases do we apply EKD?” vs. “What are the cases of application of EKD?” vs. “When is EKD applicable?”;
- words that are not ordinary language synonyms, but act like synonyms in a particular context: “Why is EKD beneficial?” vs. “Why do we use EKD?”;
- generalization and specialization of a concept (not common).

Although choosing the synonyms and grammatical forms is a creative manual process, assistance of computerized dictionaries is possible and very welcome.

“Irrelevant” Words

“Irrelevant” words are those that do not help to distinguish individual FAQs in the database. One list of “irrelevant” words exists for all the FAQ entries in the database. The list is consulted when secondary keywords for a particular entry are selected in order to reduce the number of secondary keywords because “irrelevant” words act as secondary keywords for every entry in the database. Now and then, the list gets updated considering all the secondary keywords in the database.

In a certain context, words from the list may be used as primary keywords, as it was discussed previously.

The following types of words are considered irrelevant:

- words that are not meaningful per se: “the”, “which”, “that”, “of”, etc.;
- common verbs: “is”, “do”, “can”, “exist”, “supposed”, etc.;

- all pronouns: “I”, “you”, “it”, etc.;
- common adjectives and adverbs: “common”, “typical”, “usually”;
- indicators of quantity or quality: “one”, “two”, “three”, “few”, “little”, “many”, “much”, “lot” (“a lot of”), “lots” (“lots of”), “good”, “bad”, etc.;
- words in typical phrases, like “what do you know about…”, “what kind / type of…”, etc.
- words that are common in the context of the entire information in the database: “EKD”, “enterprise”, “business”, “technology” (remember – these words are ignored in a user’s question unless required by primary keywords);
- single English alphabet letters: “a”, “b”, “c” … “z”;
- other words that do not emphasize the differences between individual FAQs within the context of the FAQ set.

The actual list of “irrelevant” words of the EKD QA System is presented in Section 4.6.3.

3.2.3 Testing the Entry

After the administrator of the system has added a new FAQ entry or updated existing one, he or she should test it. The testing is done by asking questions to the system as usually users do, yet with the focus on included or possibly not included primary and secondary keywords. According to the experience with the EKD QA System, testing may reveal that:

- the entry is good;
- the keywords are misspelled;
- there exists another formulation of the asked question which is not recognized by the entry, therefore new keywords must be added or another auxiliary entry needs to be created;
- there is a keyword conflict between two or more FAQ entries, i.e., other entries respond to the question asked in order to test this one.

According to the current experience with the EKD QA System, keyword conflicts are not frequent. If such a conflict occurs, the keyword selection of each auxiliary entry involved in the conflict must be reconsidered. Possibly, the entry may be split into two auxiliary entries with a narrower keyword selection for each one.

3.2.4 Human Resources Involved in Creating FAQ Entries

Not everybody is able to administrate the system and create FAQ entries for Prioritized Keyword Matching. The administrator of the system must possess certain skills:

- understand the knowledge domain of the FAQ answering system;
- understand the principles of how FAQs are retrieved;
- have a good command of English, if this language is used in FAQs.

Understanding of the knowledge domain and the principles of automated FAQ retrieval lets the administrator select representative auxiliary entries and keywords, which is important in order to ensure high quality of the retrieval. When creating an entry for the EKD QA System, the author of this thesis considers the text of the FAQ being created and its answer, other FAQ entries in the database, previously asked users' questions recorded in the transaction logs, his knowledge of literature about EKD and Enterprise Modelling, discussions with colleagues, and his own experience of work with EKD.

As it was mentioned previously, indexing of the FAQs (i.e., the work of selecting auxiliary entries and the keywords) goes hand in hand with the work of formulating these FAQs and finding their answers. Therefore, although a certain degree of automatization of the indexing process is possible, complete elimination of manual work is not rational because of lost quality of reasoning. Today computers are still less intelligent than humans, in average.

The advantage of Prioritized Keyword Matching with respect to involved human resources is that the total amount of work with it is reasonably small. The EKD QA System was created and is maintained by one student. Within a few months, the first version of the data structure and the algorithm of Prioritized Keyword Matching were designed, and the program code written. The most time consuming activity is to obtain a sufficient number of FAQs and to find their answers in order to populate the database.

Development of other FAQ answering systems may take more resources. The FAQ Finder project was supported by the University of Chicago Computer Science Department; the technical report [Burke et al. 1997-b] has six co-authors, which indicates that the system was developed by more than one person.

3.3 Principles of Lexical Parsing for Prioritized Keyword Matching, Multiple Lexicon

After the activities of lexical parsing of FAQs have been discussed in Section 3.2.2, let us summarize the mechanism of lexical parsing for the Prioritized Keyword Matching technique.

In computerized natural language processing, a lexicon (alias dictionary, thesaurus) is the main source of linguistic knowledge for the system. A lexicon describes real world concepts and relationships between these concepts. A concept is represented by its word(s), meaning, cases of possible use, grammatical aspects. The relationships between concepts are generalization or specialization, antonymy,

synonymy, homonymy, etc. The knowledge in lexicon has three levels of specificity: general, domain-, and application-specific [Burg 1997, pp. 80-83]. Lexicon is a must for semantic language processing; lexicon improves tremendously the methods based on keyword comparison.

3.3.1 Multiple vs. Single Lexicon

The FAQ entries in the database of the EKD QA System, once created or updated, are static. The keywords of each FAQ are known long before any users' questions are asked. Therefore the synonyms and grammatical forms of each keyword are put into the entry along with the keyword. No external source of lexical information is used during the matching of a user's question to this entry. Lexical parsing of the keywords in the entry is done *before* these keywords are used.

One may think that the EKD QA System uses no lexicon. This is not true. Each FAQ entry has a small lexicon that implements one function – identification of mutually exchangeable words (synonyms and their grammatical forms) for every keyword within the context of the given FAQ. The EKD QA System uses a multiple lexicon assembled from numerous independent small lexicons, where each of them is attached to its own FAQ entry. Figure 3.6 illustrates the difference between multiple and single (typical) lexicons.

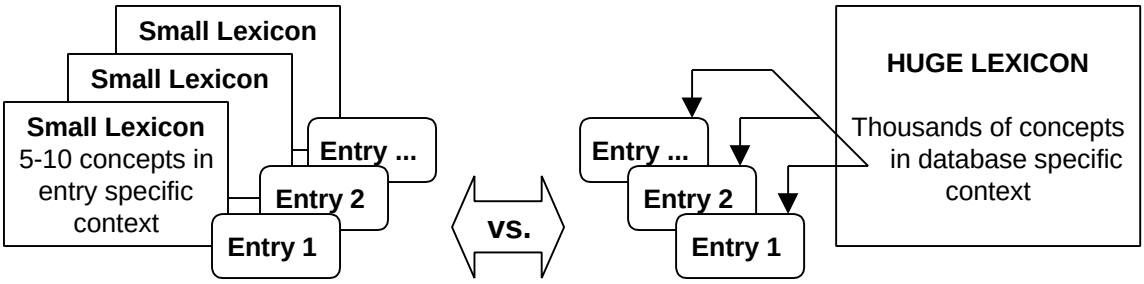

Figure 3.6 Multiple (many small units) vs. single (one large unit) lexicon.

The principal difference between both types of lexicons is in the size and *context* of autonomous units. A single lexicon has one large unit containing thousands of concepts within the context of the entire database, whereas a multiple lexicon has many small autonomous units containing 5-10, or even less, concepts within the context of one database entry (i.e., the context of one FAQ in case of the EKD QA System).

Traditionally in natural language processing, lexical parsing is applied to a user query during the text retrieval process by using a single lexicon within a database specific context. In case of a multiple lexicon used by the EKD QA System, lexical parsing is applied to the text in database entries, not user queries, long before any queries are submitted. This leads to the advantages often impossible with a single lexicon:

- Semantic relationships between words may be entry specific, like in “How do we develop a Business Process Model?” vs. “How do we acquire a Business Process Model?”, where “develop” and “acquire” are not general synonyms but are exchangeable in this particular context. It is not possible to include context specific relationships in a single lexicon.
- It is difficult to create and maintain a large lexicon. If new concepts are entered into such a lexicon or relationships between existing concepts are modified, it is difficult to adjust these changes to existing contents of the FAQ database and the lexicon. Lexical parsing within a narrow entry-specific domain takes less effort and is less error prone.
- Since lexical parsing in case of a multiple lexicon is done before any user query is submitted, the query processing is reduced to keyword matching, which requires simpler data structure and is faster than lexical analysis by using a single lexicon.

Another advantage, which is mentioned as a benefit of the use of shallow language understanding in [Whitehead 1995, p.141], is that mistakes made in one unit of a multiple lexicon do not affect other units of the same lexicon unless the erroneous unit is reused.

The drawback of a multiple lexicon in the current implementation of the EKD QA System is redundancy. Every time some keyword appears in several entries, it is written down in various forms again and again, where some of these forms are overlapping and some are not. Rewriting, however, is a minor drawback since the entries are reusable and the text editing copy-and-paste manipulation is available. There is no big waste of computer disk space either. If the database has 300 entries with 10 concepts each, there are only 3000 concepts in the database, which is less than in a small dictionary. It is possible to reduce redundancy by joining common parts of autonomous units providing appropriate links to these parts from each particular context. The trade-off between redundancy, complexity of the data structures, and access speed is a matter of implementation. We should not, however, reduce redundancy on expenses of the accuracy of the representation of FAQs.

The idea of multiple lexicon is analogous to that of object-oriented programming (OOP):

- *Definition of a lexicon* (in OOP – definition of a class). The necessary functions of lexical parsing – resolving of synonyms, grammatical forms, generalization, specialization, etc. – are defined (in OOP – definition of methods). Empty concept slots are used (in OOP – definition of attributes) since the actual concepts are not known yet.
- *Instances of the lexicon* (in OOP – instances of the class, objects). Within a series of different contexts, the concept slots, which were created during the

definition of the lexicon, are filled by the actual concepts. Now there is a series of analogous independent lexicons where each lexicon may contain five concepts as easy as five thousand.

A context sensitive multiple lexicon seems most effective with English and similar languages where the grammatical diversity of words (gender, cases, plural or singular forms, etc.) is narrow, and where it is not common to merge several words into one. This relative simplicity of English permits limited morphological analysis of words suitable within the boundaries of an autonomous unit of the lexicon.

3.3.2 Creating vs. Using an Entry

Introduction of a multiple lexicon into the EKD QA System was motivated by the following:

- these are two distinct tasks to create an FAQ entry and to retrieve it; therefore
- the workload of lexical parsing of keywords is shifted from the process of retrieval of an entry to the process of creating this entry.

This helps to keep things simple. It is *not the task of the Prioritized Keyword Matching algorithm to perform lexical parsing of keywords* in an FAQ entry. Rather, this is the task of the system's administration tool to provide the administrator with assistance to ease this process while the entry is being created. This:

- enables human reasoning and, thus, semantic analysis of the context when appropriate synonyms and grammatical forms are selected (it is difficult to computerize this task); and
- allows reducing the task of semantic matching of a user's question to an FAQ in the database to keyword comparison, which enables a simpler database structure and matching algorithm.

3.4 Benefits of the Use of Shallow Language Understanding in the EKD QA System

This section summarizes the experience of the use of shallow language understanding in the EKD QA System, with respect to the experience published in [Whitehead 1995]. The system does not attempt to understand the content of a user's question at a deep, semantic level in order to find the corresponding FAQs in its database. Query processing is more like text retrieval than traditional natural language understanding. Nonetheless, shallow language understanding is not only practical and effective, but it can also create the illusion of intelligence, especially if the scope of the domain is narrow, the database is densely populated and the system has some language processing skills such as ability to resolve synonyms.

A natural language processing technique that uses shallow language understanding has the following benefits:

- It is simple. The technique used by the EKD QA System can be implemented in a single database table with only a few columns. The query-entry matching process is fast.
- Information in the database is split into independent pieces:
  - Changes or errors in one piece have no effect on other pieces [Whitehead 1995, p. 141].
  - The information in the database can be augmented gradually, piece by piece, often by nonprogrammers [Whitehead 1995, p. 141].
  - A multiple lexicon can be used, which enables context dependent lexical analysis.

The following features make the approach of shallow language understanding more effective:

- Questions on the same topic tend to use the same words. This makes keyword matching easier [Whitehead 1995, p. 141].
- Keyword matching becomes more effective after elimination of common words like ‘is’, ‘a’, ‘the’, etc. [Whitehead 1995, p. 141]
- This research shows that, unlike in [Whitehead 1995], the interrogatives ‘what’, ‘why’, ‘how’, etc. are of little help. There are too many ways to ask the same question.
- Auxiliary entries cover most of conceivable forms of an FAQ.
- English is advantageous for shallow language understanding because the relative morphological simplicity of English words allows shifting language processing efforts from profound grammatical analysis of the text to keyword matching.
- Keywords are made of natural language alphabet letters with no distinction between lower and upper case letters. Use of digits and special characters would make the keywords too specific.

Shallow language understanding has its limits: it is difficult to resolve pronoun references (‘it’, ‘that’, etc.), to maintain complex questions, and to keep track of the context within a series of questions [Whitehead 1995, p. 141]. Nonetheless, the simplicity of this approach makes the work of setting up a new FAQ answering system easier.

4 Implementation and Operation of the EKD QA System

This section deals with implementation and operation of the EKD QA System. The implementation architecture and software components of the system, as well as maintenance of the FAQ set are discussed.

4.1 Implementation Architecture and the Software Environment of the System

As it is already stated in Section 2.2 and showed in Figure 2.5, the EKD QA System is designed considering the client-server architecture of Web-based applications.

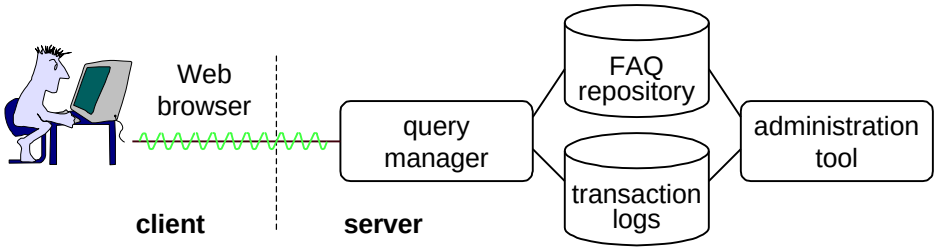

Figure 4.1 Implementation architecture of the EKD QA System.

Figure 4.1 shows the main software components of the system:

- *User interface tool* is a Web-browser on the user's computer. User input is discussed in Section 4.2.
- *FAQ repository* is the collection of FAQ entries. The data structure and maintenance of the repository are described in Section 4.3.
- *Query manager* is the main actor of the EKD QA System which processes the user input, searches through the FAQ repository, creates the output, and logs the transactions. The query manager is discussed in Section 4.4.
- *Transaction logs* are records of dialogs between users and the system. Motivation and structure of such a record has already been discussed in Section 2.2. The implementation details are presented in Section 4.5.
- *Administration tool* is used for handling the data in the system's database. The tool is not Web-based and is not available to the users of the system. The implementation of this tool is discussed in Section 4.6.

The server side operating system is Microsoft Windows NT 4.0. The query manager is a CGI (Common Gateway Interface – a standard for external programs to interface with information servers such as HTTP servers) script linked to an HTTP server. The administration tool is not connected to the Internet and uses its own graphical user interface. The query manager and administration tool are

written in Borland Delphi Pascal; they use an InterBase Server to manipulate with two dBase data tables. The user-system transactions are logged in a plain text file. The server side system communicates data to the user's Web-browser as HTML documents which may contain JavaScript code for preliminary user input validation. The front page of the system and the manual on how to use the system are the only pre-stored documents. The rest of HTML text is generated upon request.

The choice of the programming language was motivated by the following:

- the programming language has to support database management instructions;
- the programming language has to support development of both console applications (e.g., the query manager, which uses exclusively the operating system's standard input and output) and applications with own graphical user interface (e.g., the administration tool);
- the compiler, text editor, documentation, etc. for the programming language must be available at the department where the EKD QA System is developed.

Since the EKD QA System is a research prototype rather than a commercial product, the issue of independence of the server side software from the operating system was not considered important.

The dBase format of data tables was chosen because it is simple and supported by off the shelf tools such as Microsoft Access. The latter diminishes dependence of the data storage on a particular administration tool.

4.2 User Input

The following aspects of the system's user interface are discussed in this section: the layout of the input page, the technical requirements of the user input, and the system's assistance during the user input process. Input data for each type of query and the corresponding output are described in Section 4.4.

4.2.1 Layout of the User Input

Figure 1.2 in Section 1.3 shows an imaginary question input to the system. Figure 4.2 shows the layout of the real front page of the EKD QA System. This is the only page that asks for user input. It is one of the two pre-stored HTML documents at the system's disposal.

Goals 5.1 to 5.4 in Figure 2.2 motivate the input components. On the front page, the gray area is for question answering and keyword-based search, the noncolored one is for sending comments, a hyperlink allows "listing all the available FAQs".

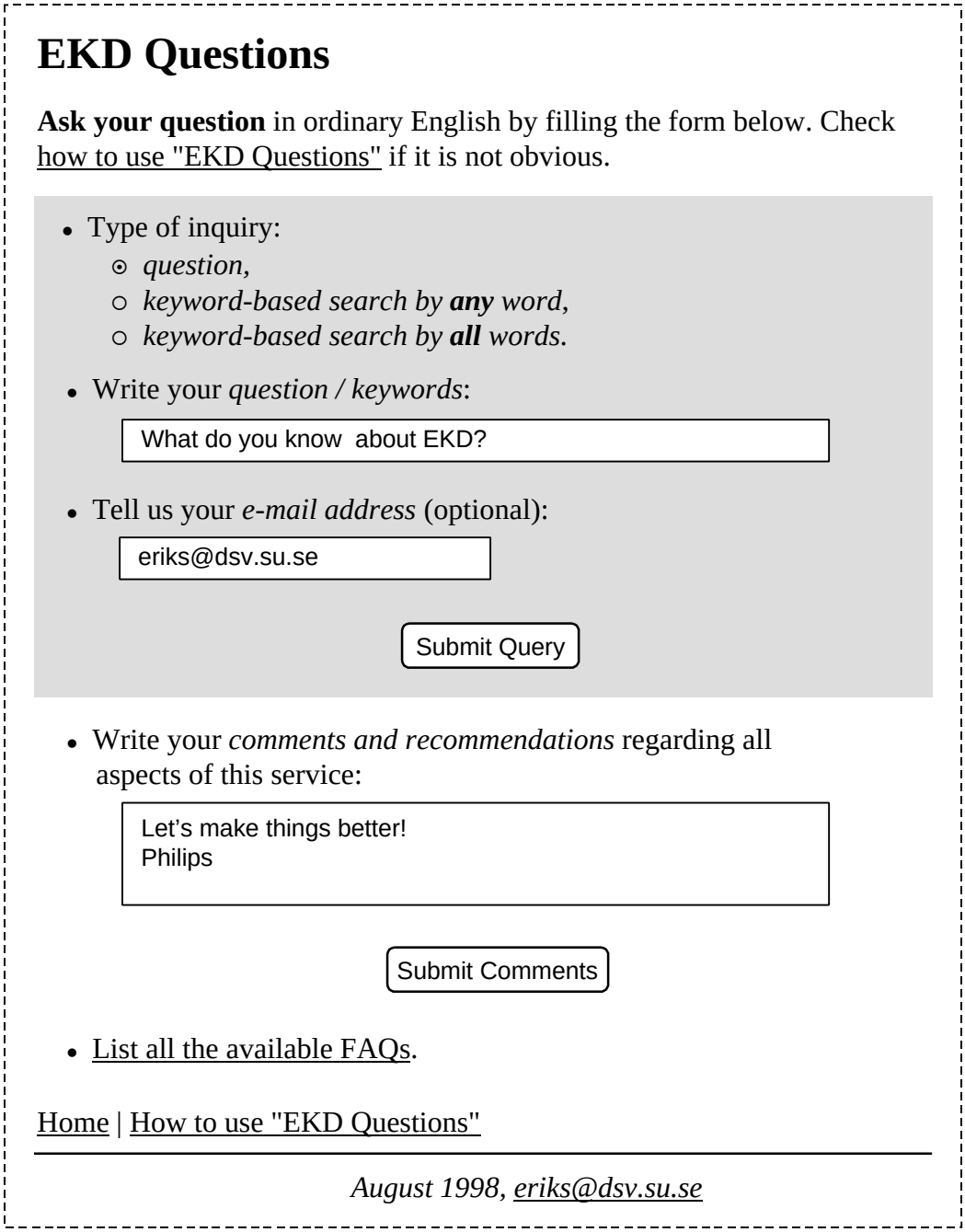

Figure 4.2 Front page of the EKD QA System.

4.2.2 Technical Requirements of the User Interface

Technical requirements are the following:

1. The system is active and available nonstop.
2. The system's user output may contain only HTML 2.0 with Cascading Style Sheets[16] Level 1 and JavaScript 1.0.

[16] http://nowheres.hypermart.net/, valid in November 2001

3. The query manager should processes a user query within 10 seconds.
4. A user may formulate natural language queries in English only.
5. The system should recognize only words with correct spelling.
6. The system should perceive text as a sequence of words separated by delimiters. A word is a sequence of the following characters: ‘A’…‘Z’, ‘À’, ‘Á’, ‘Â’, ‘Ã’, ‘Ä’, ‘Å’, ‘Æ’, ‘Ç’, ‘È’, ‘É’, ‘Ê’, ‘Ë’, ‘Ì’, ‘Í’, ‘Î’, ‘Ï’, ‘Ð’, ‘Ñ’, ‘Ò’, ‘Ó’, ‘Ô’, ‘Õ’, ‘Ö’, ‘Ø’, ‘Ù’, ‘Ú’, ‘Û’, ‘Ü’, ‘Ý’, ‘Þ’ in upper and lower cases, as well as ‘ß’ and ‘ÿ’. All other characters are delimiters.

The 1st rule is a consequence of Goal 7, the 4th – of Goal 11 in Figure 2.1. The 5th rule is imposed by limited capabilities of the EKD QA System as a research prototype.

4.2.3 System’s Assistance During the Input

The front page begins with a prompt right beneath the heading: “Ask your question in ordinary English by filling the form below.” Although the user input is simple and the use of the system seems obvious, it happened that some users did not understand what they were supposed to do. There is a brief guide ‘How to use “EKD Questions”’ available (Goal 5.5 in Figure 2.2); people, however, do not read manuals if they are not interested in the subject. The guide is second of the two pre-stored HTML documents at the system’s disposal (first is the front page). It gives description of the input items and interpretation of possible output.

When switching from keyword-based search to question answering, users often forget to set the “type of inquiry” button to “question”. In this case the question is submitted as a set of keywords. The response to such a query is a long list of FAQs not related to the asked question, and the user is confused. Therefore the input form has embedded JavaScript code which checks whether or not keywords contain the words typical for questions: “is”, ”are”, ”was”, ”were”, ”be”, ”been”, ”how”, ”where”, ”what”, ”when”, ”which”, ”who”, ”why”. If such a keyword is discovered, the user is alerted, since most probably his or her intention was to submit a question rather than keywords.

Both the manual and keyword checking were introduced after the system had been tested for a while.

4.3 FAQ Repository

4.3.1 Data Table

The FAQ repository is a single data table where each record represents an auxiliary FAQ entry (see Section 3.2.1 for description of auxiliary entries). In order to represent a complex FAQ, several auxiliary entries and, therefore, data records may be needed.

The data structure of the table is derived from the concepts showed in Figure 3.3. Description of the columns in the data table follows.

Name	Data Type	Description
ID	Numeric(9)	Unique identifier of this auxiliary entry.
QUESTION	Character(254)	Formulation of the FAQ for this auxiliary entry (concept "FAQ" in Figure 3.3).
ANSWER	Memo	Answer to the FAQ. If there are several auxiliary entries, only one has the answer recorded; others have this field empty.
PARENT	Numeric(9)	If there are several auxiliary entries, PARENT indicates ID of the entry where the answer is recorded. PARENT is 0 if this entry itself has the answer recorded.
PRIMARY	Character(254)	Primary keywords of this auxiliary entry.
SECONDARY	Character(254)	Secondary keywords of this auxiliary entry.
FOREIGN	Numeric(1)	Limit of nonmatching (foreign) words for this auxiliary entry.

The following table presents sample data in an FAQ entry, one of two auxiliary entries for "What are the links within Business Goal Model?"

Name	Data
ID	1033
QUESTION	What are the links within Business Goal Model?
ANSWER	The link types between the components of the Goals Model are: • *supports* relationship, that is essentially seen as "vertical", i.e., it is used to refine or decompose goals or other components. • *hinders* relationship, that is used to show negative influences between components of the Goals Model, and can be considered as opposite to "supports". • *conflicts* relationship, that is used to define situation when an achievement of a goal is in conflict with another. For example, a library's goal of attracting more clients by staying open longer, will be in conflict with the goal of saving money on employee wage costs.
PARENT	0

PRIMARY	goal goals; relation relations relationship relationships relate relates related relating dependence dependencies depend depends depended depending connection connections connect connects connected connecting link links linked linking
SECONDARY	components component problem problems cause causes constraint constraints opportunity opportunities support supports hinder hinders conflict conflicts sub model models submodel submodels
FOREIGN	0

Keywords are recorded as a string of words separated by delimiters (normally a whitespace). Groups of synonyms are separated by semicolons. Secondary keywords do not need synonym grouping.

The experience shows that a simple numeric representation of entry identifiers is not convenient. We can insert only a small number of new entries between two other entries if they are identified by close integer numbers. From the beginning, the entries of the EKD QA System were numbered by step 10: 1000, 1010, 1020, … Nonetheless, such a step proved too small for semantic grouping of the entries as new FAQs appeared in the database. The problem can be solved by introducing a better-structured identifier.

The experience shows that the date of creating an entry needs to be recorded for the sake of evaluation of the system's performance. The transaction logs record the time of asking a question. Nonetheless, we do not know whether the corresponding FAQ entries, if any, existed at that moment or were added later.

4.3.2 Possible Restructuring of the FAQ Repository in Order to Speed up the Query Processing

This section gives a *possibly useful* solution of hastening the query processing. It is possible to reduce the number of FAQ entries processed by Prioritized Keyword Matching by enhancing the data structure of the FAQ repository.

Primary and secondary keywords are enclosed in each FAQ entry as two strings of words separated by delimiters (columns "PRIMARY" and "SECONDARY" in the data table of the FAQ repository). While answering a user's question, the system must perform exhaustive search by Prioritized Keyword Matching through all the FAQ entries in the repository in order to be aware of every keyword. In order to reduce the number of entries for exhaustive search, let us first augment the data structure of the FAQ repository by one more data table as showed in Figure 4.3.

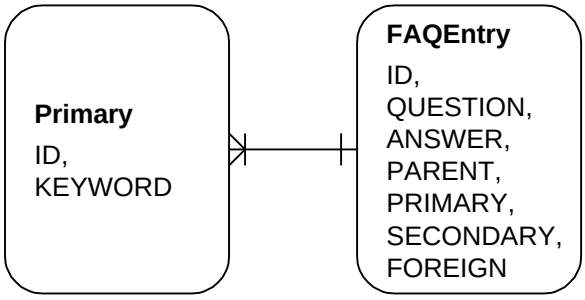

Figure 4.3 ER-diagram of the modified FAQ repository.

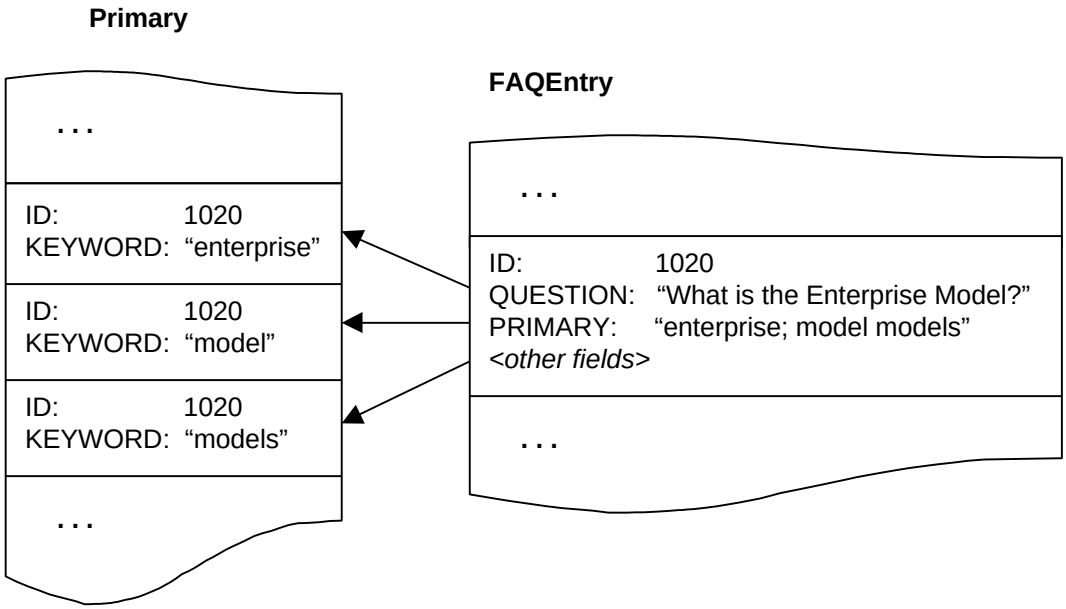

Figure 4.4 A sample FAQ entry and its primary keywords in the modified FAQ repository.

FAQEntry is the data table as described in the previous section. *Primary* is a new table. A copy of every synonym of every primary keyword from every FAQ entry is stored separately in the column *Primary.KEYWORD*. The FAQ identifier *ID* establishes the links between the records in both tables. A sample FAQ entry is showed in Figure 4.4.

In the single column *Primary.KEYWORD*, all primary keywords can be easily and rapidly accessed by means of database management instructions.

After a user submits a question, the system splits it into separate words: w_1, w_2 … w_n. Knowing these words, the system can apply them as a filter to the *Primary* table:

```
Primary.Filter = 'KEYWORD=w1 or KEYWORD=w2 or … KEYWORD=wn';
```

This is an instruction in Delphi Pascal. Since now and until the filter is removed, *Primary* contains only those records where *Primary.KEYWORD* is represented among the words of the user's question. Hence, *Primary.ID* identifies only those FAQ entries which has at least one primary keyword represented among the words of the user's question. These entries are a subset, presumably a small subset, of the FAQ repository. The system has to perform exhaustive search by Prioritized Keyword Matching only through the entries in this subset.

The modified structure of the FAQ repository as showed in Figure 4.3 is not implemented in the current EKD QA System because exhaustive search through the entire FAQ repository is still reasonably fast. In the future, however, more complex language processing may be introduced, which could increase the processing time. Then exhaustive search through the entire FAQ repository may become unacceptably long.

Another alternative to speed up the query processing is to embed FAQ entries into the program source code. It is possible because for users FAQ entries are read-only. In this case the system would not need a database engine and the system's performance would be faster. The data administration, however, would become more complicated.

4.3.3 Requirements of the Maintenance of the FAQ Repository

There are two "soft" rules set forth to the administrator regarding the FAQ repository of the EKD QA System:

- The administrator updates the content of the FAQ repository as discussed in Sections 4.3.4 and 4.3.6.
- In case of a genuine inquiry (i.e., if the question is not asked for the purpose of testing or sheer curiosity), after the system reports an unanswered or incorrectly answered question, the administrator answers it by e-mail if the address is known.

The next sections discuss maintenance of the FAQ repository.

4.3.4 Creating the Initial Set of FAQs

Before the EKD QA System was put into operation, the initial set of FAQs was created. Some critical mass of FAQs must be obtained; otherwise a question answering system is useless. We cannot expect people to contribute to the system by asking their questions if there is going to be no answers.

Two main information sources for acquisition of first FAQs for the EKD QA System were:

- examining the literature on EKD;

- inquiring academic and business people who develop and use the EKD approach in their work.

Most initial FAQs for the EKD QA System were created examining the EKD User Guide [EKD 1998]. The approach was to study the guide and to pick out the core concepts and ideas from each section with respect to importance of the section. This resulted in a number of "what is this" type of questions. The EKD QA System was put into operation after about 45 FAQs were created in the repository.

4.3.5 Formulation of an FAQ and its Answer

After the idea of an FAQ is picked up, the administrator formulates the FAQ and its answer following the guidelines:

- The FAQ should be concise, with one main idea even if it deals with a number of concepts. Several related FAQs can be created if necessary.
- The answer should highlight the essence of the FAQ. If there are several related issues to be explained, then several FAQs should be introduced. Often creation of one FAQ causes creation of another one.
- An FAQ rarely is standing alone. Therefore references to related FAQs or other information should be given in the answer.

For the EKD FAQs, the main source of the answers is EKD related literature, preferably computer-based literature. The latter enables writing the answers by the copy-and-paste method with editing afterwards.

4.3.6 Updating the FAQ Repository

As the EKD QA System is running, the transaction logs are the main source of new FAQs. The logs contain all the users' questions, as well as the information on how well the questions were answered. If a user's question has not been correctly answered, it is considered to be included into the FAQ repository by adding new entries or updating existing ones. Thus the system's competence grows step by step.

The administrator of the system should answer users' questions by e-mail if the system fails to do that. Hence, the efforts of answering a question by e-mail can be combined with the efforts of creating or updating FAQ entries in the repository.

In order to improve the content of the FAQ set, the administrator of the EKD QA System:

- collects the questions asked to the EKD QA System;
- predicts new questions by scanning the EKD related literature;
- asks the EKD developers and users to share their experience of introducing it in companies;

- analyzes the obtained questions and incorporates them into the FAQ set.

The following driving questions may help analyzing users' questions in the transaction logs:

– What was the question? Was there any answer? If yes, was it good or bad?

– Is this question frequent?

– What was the next question of this user? Was it related to the previous one? Was the previous answer sufficient?

– Does a certain sequence of questions repeat from user to user?

There exist three procedures of incorporation of a question into the FAQ repository:

- a brand new FAQ entry is created (the most often procedure);
- some new keywords are added in an existing FAQ entry;
- an existing FAQ is reformulated and its entry is split into several auxiliary entries.

The FAQ repository will never be complete because the universe of conceivable questions of any reasonably broad topic is infinite. During the operation of the system, new FAQs will be added and old ones edited in order to answer users' questions properly. Hence, the system is evolving, as stated by Goal 14 in Figure 2.1.

4.4 Query Manager

The query manager – a CGI script linked to an HTTP server – processes five possible types of user queries identified by Concepts 1.1 to 1.5 in Figure 2.6 and returns the corresponding replies identified by Concepts 2.1 to 2.4 in Figure 2.7 (submission of comments yields a simple "thank you").

Data are submitted from a Web-browser to the HTTP server on the system's host by the POST (question, keywords, comments) or GET (request for all FAQs, identifier of a particular FAQ) methods. The server starts execution of the query manager and passes the data to its standard input (POST method) or command line parameters (GET method). The query manager processes the input, through its standard output returns the reply to the HTTP server, records a transaction log, and finishes execution. The HTTP server transmits the reply back to the Web-browser.

The functionality of the EKD QA System stated in Section 2.3.2 is the functionality of the query manager. The rest of this section explains the implementation of these requirements.

4.4.1 Question Answering

In order to ask a question, a user:

- sets the type of inquiry to “question”,
- writes the question,
- optionally writes his or her e-mail address

in the gray area of the input form (Figure 4.2), and submits the query. After the query manager obtains the question, it performs exhaustive search through the FAQ repository in order to find the corresponding FAQs as explained in Section 3. The query manager answers the question by the following output:

- *User’s own question.* The user can see his or her own question with possible misspellings and judge to which extent the provided FAQs agree with the original question.
- *Corresponding one or several FAQs and their answers,* if any. The corresponding FAQs are those that match the question according to the Primary Keyword Matching technique. An example of such a reply is showed in Figure 1.3.

A message that no corresponding FAQs have been found is returned if this is the case. Also, the system informs the administrator about such an occasion by e-mail.

- *Related FAQs,* if any. The notion of related FAQ is defined in the end of Section 3.1.3. For example, the following list of related FAQs is replied to the question “What is the relationship between goals and processes?”:
  - What are the links within the Business Goal Model?
  - What is (external) business process? => What are the components of the Business Process Model?
  - What is goal / problem / cause / constraint / opportunity in the Business Goal Model?

  Each FAQ in the list has a hyperlink to its answer as explained in Section 4.4.4.

4.4.2 Keyword-Based Search

In order to submit the keywords, a user:

- sets the type of inquiry to “keyword-based search by *any* word” or “keyword-based search by *all* words”,
- writes the keywords,
- optionally writes his or her e-mail address

in the gray area of the input form (Figure 4.2), and submits the query. After the query manager obtains the keywords, it performs exhaustive search through the FAQ repository where for each auxiliary entry it matches the keywords to the text

of the FAQ and to the primary keywords considering their synonyms and grammatical variations. Matching of whole words, not substrings, is performed.

The output of the keyword-based search is:

- user's own keywords;
- a list of FAQs linked to their answers like in case of related FAQs (see the previous section), or an appropriate message if no matching FAQs have been found.

The sequence of FAQs in the list is determined by their order in the repository. The list may contain different formulations of the same FAQ, i.e., several auxiliary entries referring to the same FAQ answer.

4.4.3 Listing All the Available FAQs

By clicking on the hyperlink "List all the available FAQs" at the bottom of the front page (Figure 4.2), a user can retrieve the list of all auxiliary FAQ entries available in the database. The URL is

```
http://ekd.dsv.su.se/scripts/faq.exe?browse%3Dall
```

where "%3D" is the encoding of "=" required by the HTTP server. Upon a click, the request "browse=all" is submitted. The list retrieved this way is analogous to the lists of related FAQs and result of keyword-based search, as described in the previous two sections.

4.4.4 Retrieval of a Particular FAQ and its Answer

In order to obtain the answer of an FAQ in a list of related FAQs, result of keyword-based search, or all available FAQs, a user clicks on the hyperlink associated with the FAQ. The URL contains the identifier of the FAQ. For instance,

```
http://ekd.dsv.su.se/scripts/faq.exe?browse%3D1001
```

is the URL for the FAQ no. 1001. Upon a click, the request "browse=1001" is submitted. The system's output is the requested FAQ and its answer, a picture similar to the reply showed in Figure 1.3.

4.4.5 Sending Comments

In the comment field of the front page (Figure 4.2), a user may write his or her remarks and suggestions regarding all aspects of the FAQ answering service, the system, as well as EKD and Enterprise Modelling. As the courtesy, but not necessarily, the user may add his or her name and / or e-mail address in the end of the comments.

The query manager forwards the comments to the administrator of the EKD QA System by e-mail.

4.4.6 A Few Programmer's Notes

This section discusses two programming issues that are important for efficient implementation of the query manager, yet may be not obvious.

Error Handling

Delphi Pascal has a powerful error handling (called also exception handling) mechanism: "try-except" statements encompass the protected program instructions:

```
try
. . .
<instructions, including subroutine calls>
. . .
except on <exception type> do
<exception handler>
end;
```

Since the EKD QA System works autonomously, it is important to use this mechanism. The entire program code of the query manager has the default error handler. If the system "crashes", the error handler takes over the control: sends an appropriate message to the user and gracefully terminates the application. Separate parts of the code may have their own error handlers.

The programmer can use the error handling mechanism to tackle undesired situations by artificially raising an exception:

```
raise Exception.Create("The message for the user.");
```

To ensure that the application releases the occupied resources – data tables, files, network connections, memory – the "try-finally" statements encompass sensitive parts of the code:

```
try
. . .
<instructions that obtain the system resources>
. . .
finally
<instructions that release the obtained resources>
end;
```

Strings vs. Strings

Delphi Pascal handles both Pascal-type and C-type strings. There are two rules of thumb concerning their use:

- Use of Pascal-style strings is simple. They are convenient if a new string is created or operations with the string are performed once.

- Use of C-type strings is more time-efficient and complicated. They are efficient if operations with the string are performed in a loop. The Prioritized Keyword Matching algorithm has many embedded loops; the processing time was reduced more than twice by replacing Pascal-type strings by C-type strings.

Words of a natural language are usually stored as a sequence of letters separated by delimiters in one string. It is more time-efficient to use substring-matching operations, if possible, rather than to split strings into words and then compare separate words.

4.5 Transaction Logs

A dialog between a user and the system is logged (Goal 12 in Figure 2.1) in order to find out the users' needs and to monitor the system's performance. The main concepts of a transaction log are showed in Figure 2.8 and discussed in the end of Section 2.2. An actual transaction log records the following information:

1. date and time of the transaction;
2. IP address of the user's computer (obtained from the temporary environment variable "remote_addr" set by the HTTP server);
3. user's e-mail address;
4. action code, one character:
   - `"q"` – question,
   - `"&"` – keyword-based search using all the given words,
   - `"|"` – keyword-based search using any given word,
   - `"a"` – listing all the FAQs in the database,
   - `"i"` – retrieval of a particular FAQ and its answer by the identifier;
5. numeric code of the result of the action:
   - `0` – request satisfied,
   - `-1` – request not satisfied,
   - `-2` – no FAQs answering the question were found (like in case of –1), but related FAQs were suggested,
   - `-9` – abnormal termination performed by the default error handler;
6. text of the user query: question, keywords, or the identifier of the requested FAQ;
7. system's reply to the query (recorded only in case of question answering or keyword-based search if the result code is 0 or –2): the identifier and text of the FAQ(s).

The logs are stored in a structured ASCII file, one log per line. In a line, the log items are written in the order as above and separated by semicolons. In the text of the items, semicolons are replaced by commas, new line and carriage return characters – by tabulation.

A few sample transaction logs:

```
8/19/98 1:48:11 PM;130.237.157.72;js@dsv.su.se;q;-2;How can
an EKD model be evaluated?;RELATED: 1010 What is model in
EKD?

9/8/98 9:31:20 PM;192.116.198.224;;q;-1;What are
metamodels?;

9/18/98 5:38:32 PM;130.237.161.234;;i;0;1032;
```

The user is automatically identified by the IP address of his or her computer. Knowing the IP address, the administrator of the EKD QA System can find out where the query comes from by using the "nslookup" command in Unix.

4.6 Administration Tool

The administration tool is used in order to update the FAQ repository and the table of "irrelevant" words for Prioritized Keyword Matching, and to browse the transaction logs. The tool is not connected to the Internet; it has its own graphical user interface.

4.6.1 Window of the FAQ Repository

This window displays the FAQ data, explained in Section 4.3.1, and allows performing the following operations in the repository:

- add, edit and delete an entry;
- locate an entry containing a given substring;
- copy and paste an entry (entries are reusable);
- print all the FAQs and their answers into an HTML file.

The layout of the input for locating an entry with a given substring is showed in Figure 4.5.

By using this facility, we can search for a substring in the formulation of an FAQ, its primary and secondary keywords, and the answer. Enabling or disabling the search targets by marks in check-boxes allows combinations of the targets.

The entry adding and editing facilities use a separate window. This window has buttons to insert certain HTML tags into the text and answer of an FAQ.

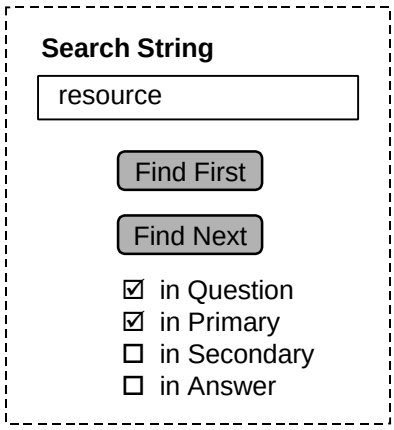

Figure 4.5 Input for locating an entry with a given substring.

Microsoft Word and Merriam-Webster OnLine

Selection of context dependent synonyms for primary and secondary keywords in FAQ entries is a creative manual process. The author of this thesis and the administrator of the EKD QA System in the same person enjoys the assistance of the thesaurus and spelling checker of Microsoft Word. Nonetheless, normally no more than $^1/_3$ of suggested general synonyms are appropriate in a particular context.

Another source of linguistic information is the Merriam-Webster OnLine dictionary[17]. It has a rich vocabulary and exhaustive explanations of words.

4.6.2 Window of the Transaction Logs

This window displays the transaction logs (see Section 4.5) which are stored in a plain text file in order to save disk space (the latter argument may be reconsidered). Nonetheless, in order to facilitate browsing, the logs are copied into a temporary data table; the window displays the data in this table.

Filtering improves readability of the transaction logs by focusing on particular records. The filtering is based mainly on the "TTable.Filter" property in Delphi Pascal. This property contains a string with a filter expression written according to certain syntax. The expression is interpreted by the database engine – it displays only those logs that satisfy the expression. The administrator of the EKD QA System writes a filter expression manually, which increases flexibility of the filtering without building a complex facility. For instance, the filter string

```
ip = '130.237.157.68' and action = 'q'
```

says that only the logs of the questions (action = 'q') asked by the given user (ip = '130.237.157.68') must be displayed.

[17] http://www.m-w.com/dictionary.htm, valid in November 2001

Another filtering facility is selecting the logs that contain a certain substring in the recorded user query and / or system's reply.

4.6.3 Window of "Irrelevant Words"

The list of "irrelevant" words for the Prioritized Keyword Matching technique is implemented in a dBase data table. This window displays the content of the table; it has facilities to add, edit, and delete words, as well as to print the entire list. The table contains the following words:

```
a able about allowed also among an and any are as at b
bad be been better between business but by c can common
commonly could d did do does doing done down e ekd em
enterprise etc exist exists f few five for four
frequently from g gave give given gives giving go going
gone good h had has have he her here hers him his how i
if in into is it its itself j k kind kinds knew know
known knows l little lot lots m many may me mean meaning
meanings means meant method methodologies methodology
methods might mine more much my n normally o of often on
one or order ought our ours out p people q r really
regarding s shall she should so some sort sorts still
such suppose supposed supposes supposing t technique
techniques technologies technology than that the their
theirs them then there these they this three thus to
together two type types typical typically u understand
understanding understandings understands understood up
upon us useful usual usually v w want wanted wanting
wants was we went were what when where which while who
whom why will with within worse would x y yet you your
yours z
```

The basis of the list was made by using the traditional stop-list creating method in Information Retrieval. A term vector of the EKD User Guide [EKD 1998] was created. By scanning the terms, words for this list were selected according to the definition of "irrelevant" words in Section 3.1.1. The administrator of the system regularly updates this list as new FAQ entries are created and new "irrelevant" words distinguished.

5 Evaluation of the EKD QA System

There are a number of independent factors that influence performance and appearance of the EKD QA System:

- *Quality of the natural language processing and FAQ retrieval technique.* In order to answer a user's question, the system must find corresponding FAQs in the repository. In order to accomplish this task, a good natural language processing and FAQ retrieval technique is needed. There exist methods in Information Retrieval to evaluate such a technique.
- *Completeness of the FAQ set operated by the system.* A user's question can be answered only if corresponding FAQs exist in the repository. There are several factors that influence completeness of the FAQ set.
- *Technical characteristics.* The most important technical characteristic is response time – how long the system processes one question. Other features are size of the system, its availability, etc.

This section discusses formal measurements of the features listed above, or informal evaluation if such is more appropriate.

5.1 Quality of the Natural Language Processing and FAQ Retrieval Technique

The quality of the natural language processing and FAQ retrieval technique of the EKD QA System is determined by the ability of this technique to select relevant FAQ entries from the repository as a question is asked. In Information Retrieval there are two parameters to measure the quality of such retrieval – recall and precision [Salton and McGill 1983, pp. 162, 164-172; Salton 1989, pp. 248-249, 277-278].

5.1.1 What is Recall and Precision in Information Retrieval

The notion of related documents (pieces of free text) is intuitive – the documents cover similar topics. An Information Retrieval system may retrieve numerous relevant and nonrelevant documents, i.e., some retrieved documents are related to the query document, some are not. *Recall* is the ability of the system to present all relevant items; it is calculated as the proportion of retrieved relevant documents among all the relevant documents in the collection. *Precision* is the ability of the system to present only the relevant items; it is calculated as the proportion of relevant documents among all the retrieved documents.

Let us visualize the situation in order to better understand it.

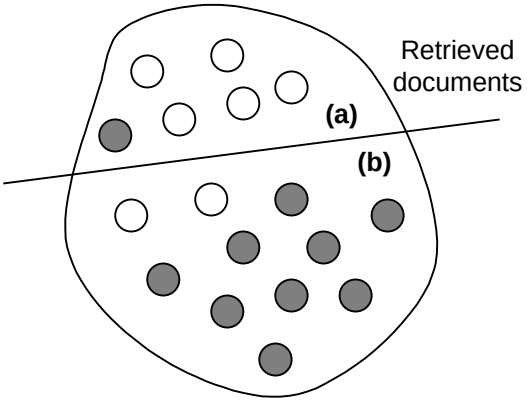

Figure 5.1 Retrieval of relevant and nonrelevant documents.

A collection comprising 17 documents is showed in Figure 5.1. Each document is represented by a circle. With respect to some query, 7 documents are relevant (light-colored) and 10 documents are nonrelevant (dark-colored). In the area (a) there are 6 retrieved documents supposedly related to the query. However, only 5 documents are relevant, 1 document is nonrelevant. Another 2 relevant documents were not retrieved and remain in the area (b).

The recall value *R* of this retrieval is measured as follows:

$$R = \frac{Number\_of\_relevant\_documents\_retrieved}{Total\_number\_of\_relevant\_documents\_in\_collection} = \frac{5}{7} \approx 0.71,$$

which is 71%.

The precision value *P* of this retrieval is measured as follows:

$$P = \frac{Number\_of\_relevant\_documents\_retrieved}{Total\_number\_of\_documents\_retrieved} = \frac{5}{6} \approx 0.83,$$

which is 83%.

For an ideal information system both precision and recall should be 100%. In reality this is true only with formal queries, for instance SQL queries of a database management system. With natural language queries, statistical Information Retrieval methods apply an empirical threshold when deciding whether to accept or reject a document. The threshold introduces a certain dependence between recall and precision: the higher precision (we want to get only relevant items), the lower recall (some probably relevant items get lost), and vice versa. In order not to loose information, usually recall is raised on expenses of precision. Query customization lets improve precision until the user is satisfied.

Measurements of recall and precision use the notion of relevant and nonrelevant documents. It is not clear how the system knows which documents are relevant and which are not. The system does not know. The system's responses must be checked

manually or using another, more reliable, system. After several measurements by different queries are done, the average recall and precision can be calculated.

5.1.2 Actual Measurements

This section describes the measurements of recall and precision of FAQ retrieval, based on Prioritized Keyword Matching, for the EKD QA System. It is important to understand that this is not a measurement of the quality of the idea of Prioritized Keyword Matching. Rather, this is an evaluation of a particular implementation of the algorithm and data structure described in Section 3.1. As the Prioritized Keyword Matching algorithm, data structures and their content are elaborated, the figures of recall and precision improve. The idea of Prioritized Keyword Matching is more complex and cannot be expressed by figures of recall and precision.

The notion of corresponding FAQ is subjective. Ideally, the correspondence between a question and the retrieved FAQ should be determined by the author of the question who knows what he or she meant, and a group of EKD experts who know the correct answer. Instead, this correspondence was determined by the author of this thesis considering his own common sense and expertise in EKD.

The measurements were based on data in the transaction logs and done manually. At the moment of taking the measurements, there were recorded more than 80 questions asked by people other than the administrator of the EKD QA System. From those more than 80, selected were question / reply pairs where the questions were not duplicate (which happened if the questions were posted for demonstration purpose) and without spelling mistakes.

Recall and precision were measured for close answers; the quality of the retrieval of related FAQs was measured separately. In order not to be influenced by completeness of the FAQ set, only those question / reply pairs were considered where the question had the corresponding FAQs in the database at the moment of asking it. In order to prove that the corresponding entry existed at that moment and, therefore, the measurement of recall is applicable, there must be the creating date recorded for each transaction log and FAQ entry. Unfortunately, such dates were not recorded for the FAQ entries. Therefore the fact of existence of each entry was determined by using human memory and comments on old printouts of the transaction logs.

Eventually, not so many transaction logs were selected. Recall and precision were measured by using 17 question / reply logs that satisfied the criteria above. One may object that such a number of records is not representative. The reason of so small number is simple. Many of the questions were asked by the developers of EKD for testing purpose. Most of the asked questions had no corresponding FAQs in the repository at the moment of asking the question, therefore they do not qualify for measuring the quality of natural language processing and FAQ retrieval. Waiting for increase of adequate users' questions would retard the submission of

this thesis, which was not considered a rational decision.

In the measurement tables presented further, the prevailing values are 0 and 1 – not typical for traditional Information Retrieval systems. There are two reasons for that. First, FAQs in the repository are organized so that they are not overlapping, therefore usually one FAQ, if any, answers a question. Second, the Prioritized Keyword Matching technique distinguishes a close answer from related FAQs, therefore it retrieves usually one FAQ, if any, as a close answer.

Two tables follow. The first one shows figures of recall and precision of *close answers retrieved as close answers* only. Such retrieval has rather high recall and very high precision.

The precision value 0.95 slightly overestimates the system's ability not to retrieve garbage. In three cases when the corresponding FAQ did not exist in the repository and the measurements of recall and precision were not applicable, the system returned nonrelevant FAQs instead of the "not found" message.

The next table shows figures of recall and precision of *close answers retrieved as either close answers or related FAQs*. Such retrieval has higher recall and still high precision.

Proportion of Relevant Related FAQs

Formal recall and precision of related FAQs was not measured because it was difficult to establish unambiguous borderline between related and nonrelated FAQs in the repository. It is much easier to observe retrieved FAQs and then judge which of them are and which are not related to the user's question ignoring the rest of the repository. The latter is not allowed in case of formal measurements of recall and precision.

40 cases were considered for the measurement of proportion of relevant related FAQs, i.e., the ratio of relevant related FAQs retrieved among all the related FAQs retrieved. Results are the following:

- in 15 cases the proportion was 1;
- in 2 cases the proportion was $^2/_3$;
- in 9 cases the proportion was ½;
- in 1 case the proportion was $^3/_7$;
- in 13 cases the proportion was 0.

The average proportion of relevant related FAQs is

$$\frac{15\times 1 + 2\times \frac{2}{3} + 9\times \frac{1}{2} + 1\times \frac{3}{7} + 13\times 0}{15+2+1+9+13} \approx 0.53.$$

Like precision, this proportion also illustrates the system's ability to retrieve or not to retrieve garbage – in average, 53% of suggested related FAQs are relevant, 47%

are not.

Table: Recall and precision of *close answers retrieved as close answers* only.

	Close answers in repository	**Retrieved**		**Recall**	**Precision**
		Actual close answers	**Total as close answers**		
1.	1	1	1	1	1
2.	1	0	0	0	Undefined
3.	1	1	1	1	1
4.	1	0	0	0	Undefined
5.	1	1	1	1	1
6.	1	1	1	1	1
7.	1	0	0	0	Undefined
8.	2	0	0	0	Undefined
9.	1	1	1	1	1
10.	1	0	0	0	Undefined
11.	1	1	1	1	1
12.	1	0	0	0	Undefined
13.	2	2	2	1	1
14.	1	1	1	1	1
15.	1	1	2	1	0.5
16.	1	0	0	0	Undefined
17.	1	1	1	1	1
			Average:	0.65	0.95

Table: Recall and precision of *close answers retrieved as either close answers or related FAQs*.

	Close answers in repository	Retrieved		Recall	Precision
		Close answers	Total		
1.	1	1	1	1	1
2.	1	0	2	0	Undefined
3.	1	1	1	1	1
4.	1	1	7	1	0.14
5.	1	1	1	1	1
6.	1	1	1	1	1
7.	1	1	2	1	0.5
8.	2	2	3	1	0.67
9.	1	1	1	1	1
10.	1	0	1	0	Undefined
11.	1	1	1	1	1
12.	1	1	1	1	1
13.	2	2	2	1	1
14.	1	1	1	1	1
15.	1	1	2	1	0.5
16.	1	1	1	1	1
17.	1	1	1	1	1
			Average:	0.88	0.85

Summary

In order to judge the figures of recall and precision of the EKD QA System, we have to compare them with analogous measurements done for other systems. There are not so many actual recall and precision figures published though.

As it was already mentioned in Section 5.1.1, usually there is a trade-off between precision and recall: higher precision of retrieval yields lower recall, and vice versa. [Salton and McGill 1983, p. 170] presents a series of recall-precision measurements for some Information Retrieval system. In order to increase the effectiveness of retrieval, the system used term vectors of "concepts" representing classes of synonyms. 35 queries were applied, the average recall and precision calculated. Figure 5.2 shows the recall-precision graph (long thin) of "that system" based on 10 recall-precision value pairs.

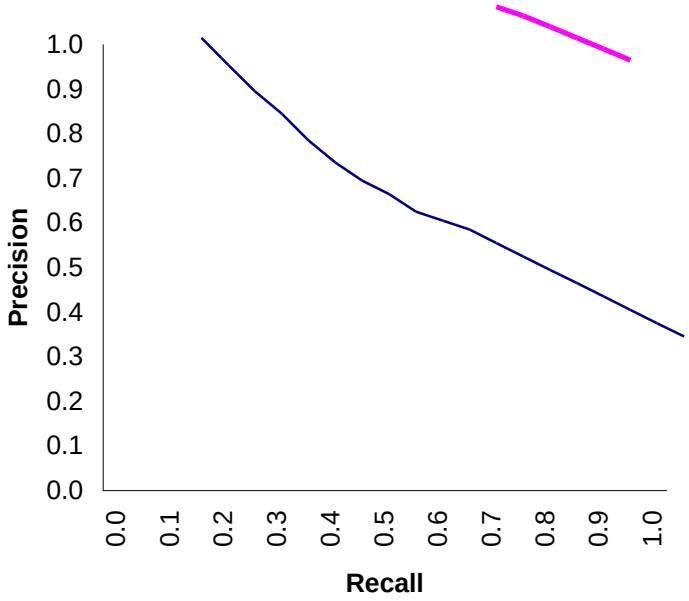

Figure 5.2 Recall-precision graphs for "that system" (long thin) and the EKD QA System (short thick).

The recall-precision graph of the EKD QA System (short thick, Figure 5.2) is based on only two recall-precision value pairs – (0.65, 0.95) and (0.88, 0.85) – where each was obtained by applying 17 queries. With the same recall values, the EKD QA System has much higher precision, as the table below shows:

Recall	**Precision**	
	"That system"	EKD QA System
0.65	≈ 0.42	0.95
0.88	≈ 0.28	0.85

[Burke et al. 1997-a] presents evaluation of FAQ Finder. There are no precision measurements for FAQ Finder. Instead, a new measure – rejection, is introduced to evaluate the system's ability not to retrieve nonrelevant items. The highest recall value for FAQ Finder is 0.68, which is close to the lowest recall value for the EKD QA System – 0.65. Nonetheless, "rejection is somewhat low for reasonable values of recall, meaning that the system [i.e., FAQ Finder] confidently returns garbage in most cases when there is no correct answer in the file" [Burke et al. 1997-a, p. 62].

Although the measurements of recall and precision for the EKD QA System may be criticized for not being representative, they shed positive light on Prioritized Keyword Matching and its cornerstones:

- *Sufficient number of auxiliary entries* helps to meet a large number of possible formulations of a user's question.
- *Good selection of primary and secondary keywords* in an entry highlights representative concepts of the FAQ. Narrower context of the keywords in a bigger number of possible contexts (more auxiliary entries) increases precision of retrieval.
- *Good context dependent controlled vocabulary, i.e., a multiple lexicon*, provides an ability to resolve grammatical forms and context dependent synonyms of keywords.

Those "sufficient and good" FAQ entries for Prioritized Keyword Matching are created applying human reasoning. Automated indexing like in Information Retrieval currently does not yield an index of so high quality as manually selected sets of keywords and controlled vocabulary.

Comparison of both recall-precision value pairs of the EKD QA System – (0.65, 0.95) and (0.88, 0.85) – brings up an unexpected conclusion. The latter is achieved by means of primary keywords only. Therefore simpler and easier-to-use keyword matching without secondary keywords and limit of nonmatching words can be implemented and still render high quality of FAQ retrieval.

Recall and precision of the EKD QA System depend on the quality of FAQ entries which, in its turn, depends on the experience of the administrator of the system and the number of actual users' questions asked, recorded, and analyzed. As the experience of the work with the system increases, the quality of FAQ entries grows, and the system evolves.

FAQ retrieval based on Prioritized Keyword Matching has no inevitable trade-offs between recall and precision. As the quality of FAQ entries grows, both recall and precision increase.

The current version of Prioritized Keyword Matching has its weaknesses – it does not recognize phrases and poorly distinguishes sentences with different meanings but similar appearance. Further research should soften this problem (see Section

6.2.1).

5.2 Completeness of the FAQ Set

Completeness of the system's FAQ set indicates whether the system has enough FAQs interesting for the users or not: a corresponding FAQ entry should exist in the repository at the moment when a user asks his or her question. Besides the quality of natural language processing and FAQ retrieval, this is the other crucial factor that influences the system's ability to answer questions.

There are several issues to be considered when we discuss completeness of the FAQ set of the EKD QA system:

- In the beginning, the system is able to answer only a few questions. We may ask a number of questions and do not get them answered, which indicates that the system has a bad ability to answer questions. After these questions are incorporated into the FAQ set, which is normally done, we may ask them once more and get them answered, which indicates that the system has a perfect ability to answer questions. Completeness of the FAQ set is ever improving and should not be measured by static figures.
- Completeness of the FAQ set is subjective with respect to the user audience. Novice EKD users ask simple questions, like "Why is EKD good?", which are normally answered. On the contrary, the developers of EKD tend to challenge the system by asking advanced questions, like "How much do you have to know about the EKD notation in order to be able to participate in a modelling session?" This question was not immediately answered, which did not imply that the system was bad.
- In case of genuine inquiries, the administrator of the system is supposed to reply to unanswered questions by e-mail and later incorporate these questions into the FAQ set. Hence, the notion of unanswered questions becomes relative.
- People may inquire about issues outside the scope of the subject of the FAQ set. For instance, once the EKD QA System received a question "What is scenario?" There were no corresponding FAQs; new one was put into the FAQ set telling that the concept of scenario is not well developed in EKD.
- It is convenient to have a textbook or manual that covers the domain of the FAQ set. We may evaluate completeness of the FAQ set with respect to such a book.
- Basically, the EKD User Guide [EKD 1998] is used to answer the FAQs in the EKD QA System. It is estimated that these FAQs cover about half of "what is this" type of issues in the user guide. Yet users do not ask questions from a book, therefore the book cannot guarantee that the FAQ set is complete.

We may calculate what is the proportion of answered questions among all the asked questions. Nonetheless, such figures do not illustrate the general quality of a

question answering system and may be misleading. The system is evolving; the FAQ set will never be complete. The universe of conceivable questions of any reasonably broad topic is infinite. The aim of the administrator of the EKD QA System, however, is to make the FAQ set as complete as possible by following the guidelines described in Section 4.3.6.

5.3 Technical Characteristics of the EKD QA System

The discipline of Software Metrics offers a number of measures of software-related activities, products, and resources [Fenton and Pfleeger 1997, Chapter 3]. A few of them were considered in order to describe the technical characteristics of the EKD QA System:

- *Availability, dependence on the operating system (OS).* The system is available nonstop to its users from any computer with WWW browsing capabilities, wherever this computer is located and whatever its OS is. The server side software is written in Borland Delphi Pascal and uses dBase data tables. Therefore the most appropriate server side OS is Microsoft Windows NT. The server side software is not OS independent, which is by far not important for the EKD QA System as a research prototype.
- *Query processing time* shows the dispatch of the query manager:
  - the usual time to match a user's question to a separate FAQ entry – one record in the data table – is 1 to 2 milliseconds;
  - the usual time to process a user's question is 0.3 to 1 second;
  - the usual time to process user's keywords is around 0.3 seconds.

Query processing takes more time if there are more records in the database. The above measurements were taken for 95 FAQ entries. The processing time is subjective because it depends on the hardware, efficiency of the compiler, skills of the programmer. The query manager of the EKD QA System is operated in the Microsoft Windows NT 4.0 environment using a Pentium 166 MHz processor and 48 MB RAM.

- *Size of the system*. Size of the program code and size of the data files are two separate measures.

	Nonblank lines of code
Query manager	977
Administration tool	1155

Only manually written lines of Pascal code were counted. The graphical user interface of the administration tool of the EKD QA System was built using an

advanced form editor which created the code automatically.

Data size (95 FAQ entries and the list of "irrelevant" words in *.dbf*, *.dbt*, and *.mdx* files) is 361 KB.

- *Maintenance of the FAQ set* is a computer assisted manual process which can be accomplished by one person who has certain skills in order to create FAQ entries as described in Section 3.2. It is easy to start operating a new FAQ answering system as the software of the EKD QA System is installed.

5.4 Brief Summary

We can distinguish more and less advantageous aspects of the EKD QA System:

- The system is advantageous from the user's point of view: it is easily available to remote users, its natural language processing technique has good figures of recall and precision, and it is quick.
- The most difficult aspect of the system is maintenance of the FAQ set because the task of creating new FAQ entries takes manual work and requires certain skills. It is not likely, however, that currently there exists an analogous system that is easier to maintain while having reasonably high quality of FAQ retrieval.

The discussions concerning different types of FAQ answering systems (Section 1.4) and the benefits of the use of shallow language understanding (Section 3.4) also illustrate various qualities of the EKD QA System.

6 Further Research and Conclusion

The development of the EKD QA System has reached the stage of a working application with almost 100 entries in the FAQ repository. There are a number of possible directions of developing the ideas implemented in the system and improving the system. Few of them are mentioned below.

6.1 Increasing the Functionality of an FAQ Answering System

6.1.1 Gap List

The idea of gap list is not new and is explained in [Whitehead 1995]. When a user asks a question that the system cannot adequately answer, the question identifies a gap in the content of the FAQ repository. The user should be able to post the question to a gap list. Everybody should be able to browse the list; the knowledge domain experts should be able to search, edit, answer, and delete gaps from the list by using an editor. This way the system could acquire new FAQs in its repository and new knowledge.

The EKD QA System does record all users' questions, including those not answered or poorly answered. Nonetheless, these questions are not publicly available, neither to expert nor nonexpert users. Implementation of the gap list as described above would make the system more open to the knowledge possessed by its users.

6.1.2 Improving the Administration Tool

The approach of Prioritized Keyword Matching requires lexical analysis of primary and secondary keywords in an FAQ entry while this entry is being created. Therefore facilities to ease this work – thesaurus, spelling checker, grammar prompter, etc. *included into the administration tool* – are welcome. Such facilities already exist. The issue is to find, adjust, and incorporate them.

Another issue is a possible ability of the tool to learn – to remember context dependent similarities of language constructions.

6.1.3 Experience Reports

The ancestors of EKD – previous Enterprise Modelling techniques – have been successfully introduced in various companies (see Section 1.2.1). However, the experience of the use of Enterprise Modelling within these and other companies is poorly gathered.

Since the EKD developers and users are located far from each other, the system's functionality could be extended by experience gathering through the Internet. This feature would be analogous to the gap filling described above, yet for bigger text

masses. The features of such a service could include easy browsing and comparing the reports, extracting particular information, discussion facilities, and version control.

6.1.4 Searching Generic Patterns

According to the basic idea of Prioritized Keyword Matching, an FAQ in the database is a pattern that identifies a class of questions with similar meanings. Primary and secondary keywords identify the concepts relevant to this pattern. Each keyword is represented by a group of synonyms. In FAQ answering, the pattern is a human language sentence, the keywords identify concepts expressed in human language, and the synonyms are natural language words. We may generalize the pattern mechanism of Prioritized Keyword Matching and apply it to social, physical, chemical, etc. phenomena. We can organize generic patterns of these phenomena like FAQs in a database and search through the database by using the Prioritized Keyword Matching algorithm.

6.2 Enhancing the Natural Language Processing Technique

6.2.1 Two More Features of Prioritized Keyword Matching

When the first ideas of Prioritized Keyword Matching appeared, the main concern of the research was good recall of FAQ retrieval (i.e., how to retrieve as many relevant FAQs from the database as possible). Precision (i.e., how to refine the retrieval so that it contains as little garbage as possible) was less important. The EKD QA System has good recall figures – 88%. There are two directions of improving precision (which is already not bad): recognition of phrases and introduction of forbidden words.

The need to process phrases is obvious – the system cannot even distinguish “modelling process” from “process modelling”. The meaning of forbidden words is less apparent. Consider the following two questions:

1. Why do companies use EKD? (*Why* is EKD used, for what reason?)
2. Which companies do use EKD? (*Where* is EKD used?)

Both questions have the same primary keywords “EKD” and “use”, the same secondary keyword “companies”, and a number of “irrelevant” words. The questions look similar, but have different meanings. The current version of the Prioritized Keyword Matching technique could distinguish both meanings by an exhaustive collection of many auxiliary entries with thoroughly selected primary keywords. We can, however, define that the words “why” and “reason” are *forbidden* for the meaning of the second question which deals with “where”. We do not introduce separate auxiliary entries to express “why”, “reason”, “which”, “where”, etc., and the system does not mix up both questions.

Hence, a new version of Prioritized Keyword Matching would distinguish 4 types of keywords: required (primary keywords), optional (secondary keywords), forbidden (a new concept), and "irrelevant" words. After the processing of phrases and forbidden words is introduced, possibly, the potential of the development of Prioritized Keyword Matching as a shallow language understanding technique will be depleted.

6.2.2 Towards Deeper Language Understanding

One of the basic entities making Prioritized Keyword Matching flexible is the notion of multiple lexicon (see Section 3.3.1). It defines wording of concepts and their relationships within the context of a particular FAQ. It would be a challenge to move forward into deeper language analysis using the nontraditional multiple lexicon. Nevertheless, we must keep in mind that semantic natural language analysis is more complicated than shallow language understanding, therefore the value of such a research may be more theoretical than practical for FAQ answering.

The first step into deeper language analysis could be categorization of primary and secondary keywords into subject, predicate, and complement. Or we better distinguish entities, relationships between them, and surrounding ("in the Process Model", "in EKD"). If the system knows who does what with whom, it can better suggest related FAQs. As well, it can give better hints on what kind of related information exists in the database. This may provoke a user to ask more questions and make his or her interaction with the system more exciting and game-like.

Further, knowing the last answered question and its context (i.e., the particular autonomous unit of the multiple lexicon), we may try to resolve pronoun references "it", "this", "that", etc., and make a kind of alive dialogue with the user.

6.3 Conclusion

Initially, this research was motivated by the idea of improving dissemination of information on WWW by answering people's questions, and a number of drawbacks of existing FAQ lists there. The biggest drawbacks are lack of feedback from users concerning their actual needs – users are not able to ask any questions, and poor user interface – often it is difficult to find information in FAQ lists.

One of the two study directions of this thesis is design and functionality of FAQ answering systems. Unlike Artificial Intelligence question answering systems that focus on generation of new answers, FAQ answering systems retrieve existing answers from their databases. Two earlier approaches to FAQ storing and retrieval were examined in Section 1.4. According to the approach of FAQ Finder, a system does automated indexing of external FAQ files where the questions are already formulated, answered, and stored. After the system receives a user's question, it performs syntactic parsing and lexical analysis of the question in order to extract concepts, and then performs concept matching in order to find corresponding FAQs

in the index. According to the approach of Auto-FAQ, followed by the EKD QA System, there is no external source of FAQs. Therefore no automated indexing of such a source is done. Instead, human beings create and store FAQs directly into the system's database. The retrieval of FAQs is based on limited lexical analysis and keyword matching.

The description of the EKD QA System begins with reasoning about various desired features of the system without considering implementation details. The Goal Model of the system has been created. The components of this model motivate the conceptual architecture and processes of the system.

The other study direction of this research is developing a natural language processing and FAQ retrieval technique. After a user has asked his or her question, the EKD QA System performs exhaustive search through its database and matches the question to each FAQ entry. The Prioritized Keyword Matching technique was developed in order to semantically compare two natural language sentences – a user's question and an FAQ. This technique fills the gap between statistical methods of Information Retrieval which work with large documents and are not effective with short sentences, and semantic natural language processing in Artificial Intelligence which is difficult. The data concepts, informal and semi-formal algorithms, as well as the process of creating FAQ entries for Prioritized Keyword Matching are described in this thesis.

A novelty in this research is the formulation of the notion of multiple lexicon which brings the issue of context into the traditional understanding of a lexicon. While one may suspect that analogous ideas have been used before, the author of this thesis could not find references to a formulation of the idea of multiple lexicon. A multiple lexicon:

- defines context specific relationships between words, which is not possible with a traditional single lexicon;
- is easy to maintain, comparing to a traditional single lexicon;
- is convenient with shallow language understanding which emphasizes keyword matching rather than profound grammatical analysis of the text.

Benefits of shallow language understanding are the following:

- The techniques of shallow language understanding are simple. The EKD QA System was created and is maintained by one student.
- The information in the database is split into independent units:
  - Changes in one piece have no effect on other pieces.
  - The content of the database can be augmented step by step.

FAQ retrieval based on Prioritized Keyword Matching, as implemented in the EKD QA System, shows good figures of the main quality measures in Information

Retrieval – recall and precision. Two obtained recall-precision value pairs are (0.65, 0.95) and (0.88, 0.85). Such a success is attributed to the cornerstones of Prioritized Keyword Matching – selection of auxiliary entries, selection of primary and secondary keywords, and context dependent controlled vocabulary provided by a multiple lexicon.

The EKD QA System is an evolving system. The FAQ repository of the system will never be complete. New FAQs are added and old ones edited in order to answer users' questions properly. The system's FAQ retrieval capability by using Prioritized Keyword Matching depends on the quality of FAQ entries which increases along with the number of questions asked, recorded, and analyzed, and with the experience of the work with the system.

As a Web-based application, the EKD QA System has enriched the power of multi-media technology on WWW by limited human language understanding.

Still, what is the main advantage of the EKD QA System? What makes the principles of the EKD QA System competitive? The answer is – *simplicity* and *high quality of FAQ retrieval*. The system was created and is maintained by one student, and it has rather good recall-precision values of FAQ retrieval. Although the maintenance of the system's FAQ set requires certain skills, we can adjust the complexity or simplicity of FAQ entries to needs of any website.

PART 3

CONTINUED EXPERIENCE IN AUTOMATED FAQ ANSWERING

Preface for Part 3

Part 3 comprises two publications which describe further development of the FAQ answering system and the ideas around it after the licentiate thesis in Part 2 was published.

"Automated FAQ Answering: Continued Experience with Shallow Language Understanding" [Sneiders 1999-b] is a rather complete summary of the licentiate thesis presented in Part 2 of this thesis. The paper introduces some new features not covered in the licentiate thesis. Primary and secondary keywords are called required and optional keywords here. Forbidden keywords are a new type of keywords. Furthermore, the paper introduces substitutes – keywords common for a number of FAQ entries – and explains how the FAQ answering system processes phrases.

"Application and Maintenance Aspects of an FAQ Answering System" [Sneiders and Larsson 2001] was written after extensive testing of the FAQ answering system had been conducted. The report does not focus on the technical details of the FAQ retrieval but rather discusses the social aspects of the FAQ answering system, such as the user interface, the application domains of the system, as well as the human resources needed in order to install and maintain the system.

The layout and pictures in this edition of the publications are adjusted to the new format and are different from those in the original publication. Internet addresses in the references are updated.

Automated FAQ Answering: Continued Experience with Shallow Language Understanding

Introduction

People sitting in front of their computers expect quick solutions. People browsing a website want to get quick answers to their questions. In order to enable the latter, an evolving WWW-based automated FAQ answering system, which provides pre-stored answers to users' questions asked in ordinary English, has been developed.

In an FAQ collection, the information supplier tries to answer in advance typical questions that the information users may have; FAQs are intended to solve certain problems. Traditionally such a collection is organized as an ordinary list, which has several deficiencies:

- *An FAQ user is not given a chance to ask any questions.* Instead, the user is forced to scan through a long list (or several lists) of various questions in order to find a question, similar to the user's own question, which may not exist in that list.
- *The information supplier does not know the actual questions that arise.* Rather, the information supplier answers possible questions in advance. Nonetheless, these possible questions do not always satisfy the users' needs.
- *FAQs in a list may be poorly organized.* If the number of FAQs is large and their order chaotic, then there are two options of navigation through the list: (1) to read all the questions or (2) to search for keywords by free-text substring search facility. It is, however, not possible to use substring search if the list is spread over several documents. It is an advantage if FAQs in a list are semantically grouped. But even in this case the grouping may be ambiguous, and a user may not know where exactly to look for a particular FAQ.
- *There may be several FAQs in a list that answer the user's question.* Some more FAQs may be related to the question. If the list is not well structured, a user has to scan through the whole list in order not to loose possibly valuable information.
- *An FAQ list may be too long, sometimes scattered over several documents.* It is difficult to locate a small piece of information in a large text mass. Often people either easily find what they want or give up.

These deficiencies pertain to FAQ lists whatever medium carries them – WWW, Usenet newsgroups, CD, paper, etc. An FAQ answering system overcomes them by retrieving FAQs upon a request expressed in natural language, and by storing the request for further analysis.

In order to give an idea of the system's functionality, an example of asking a question follows (Figure 1).

Figure 1 Example of a user question to the FAQ answering system.

The subject of the sample question is one of the techniques of Enterprise Modelling [Bubenko 1994], which is a general business planning methodology. By using a Web-browser, the user submits his or her question to the system. The system receives the question and searches through its database in order to find pre-stored FAQs that correspond to the question. The system recognizes different formulations of the question. After one or several, if any, relevant FAQs are found, the system sends them and their answers back to the user, as showed in Figure 2.

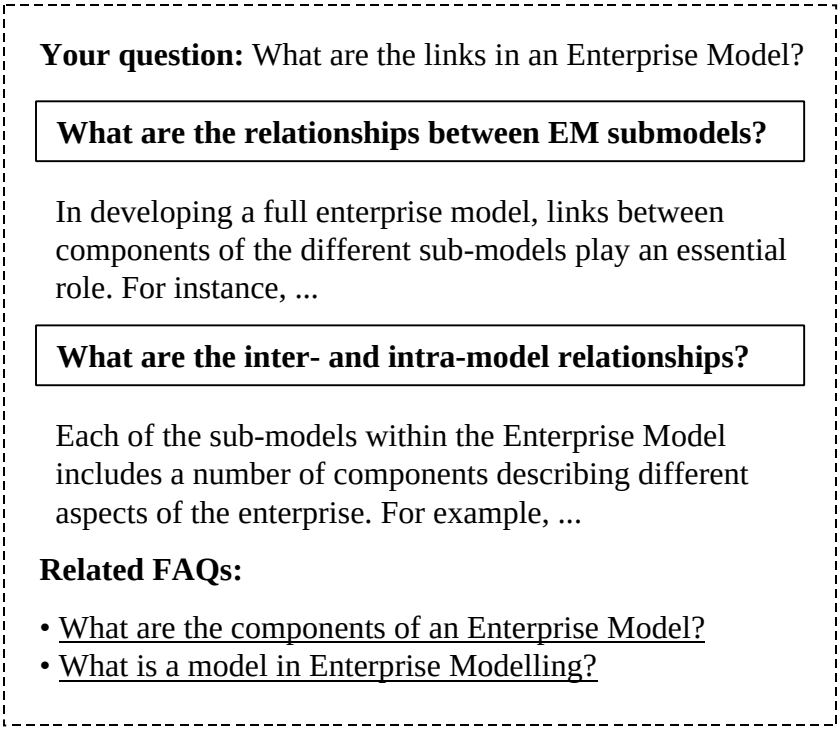

Figure 2 Reply to the question (the text is cut).

A new natural language processing technique for FAQ answering has been developed within the scope of this research. The technique is called Prioritized Keyword Matching. It uses *shallow language understanding*, which means that the

FAQ answering system does not comprehend a user question. The system formally matches the question to FAQ entries in the database; the matching is based on keyword comparison. The system performs no syntactic parsing of the question and does not extract semantic concepts. Lexical and morphological analysis of keywords, however, is done in order to enhance the language processing. The first version of the technique implemented in an FAQ answering system was introduced in [Sneiders 1998]; a thorough discussion can be found in [Sneiders 1999-a]. This paper continues the discussion with an improved version and preliminary evaluation of the technique.

Approaches and Roles of Automated FAQ Answering

Unlike Artificial Intelligence question answering systems that focus on generation of new answers, FAQ answering systems retrieve existing question-answer pairs from their databases. Two representatives – FAQ Finder and Auto-FAQ – illustrate two types of FAQ answering systems.

FAQ Finder [Hammond et al. 1995; Burke et al. 1997-a] is a system designed in order to improve navigation through already existing external FAQ collections. The system has an index – FAQ text files organized into questions, section headings, keywords, etc. In order to match a user question to the FAQs, the system (1) does syntactic parsing of the question, identifies verb and noun phrases in the question, and (2) performs semantic concept matching in order to select possible matches between the question and target FAQs in the index.

Auto-FAQ [Whitehead 1995] maintains its own FAQ set; the system does not perform indexing of an external FAQ collection. The system uses other approach to automated FAQ answering – that of shallow language understanding – where the matching of a user question to FAQs is based on keyword comparison enhanced by limited language processing skills. Question answering is more like text retrieval than traditional natural language processing.

The FAQ answering system developed within the scope of this research follows the approach of Auto-FAQ. Nonetheless, the language processing and FAQ retrieval technique is not published in [Whitehead 1995]. After a version of Auto-FAQ had been built, the development of the system stopped (according to personal communication with S. D. Whitehead).

The target system of this research is designed following several principles:

- By using the system, a user can ask natural language questions, perform keyword-based search through the FAQ collection, and browse all the FAQs in the collection.
- The system maintains its own FAQ set, as opposed to indexing an external FAQ source, and uses shallow language understanding when matching a user question to FAQ entries in the database.

- The system is evolving because its question answering ability improves as more questions are asked, recorded, analyzed, and new FAQ entries in the database created.
- The system is Web-based because WWW is a rapidly growing medium convenient for asynchronous communication and popular in the business and academic environments. Besides, a Web-browser is a ready-to-use graphical user interface tool; there is no need to build another one.

The system's server side operating system is Microsoft Windows NT 4.0. The user query processor is a CGI script linked to an HTTP server. The administration tool uses its own graphical user interface and is not connected to the Internet. Both the query processor and administration tool are written in Borland Delphi Pascal. The reasoning, architecture, functionality and implementation of the system, as well as creation and maintenance of the system's FAQ set are presented in [Sneiders 1999-a]. A version of the system is available[18].

We can distinguish the following roles of an automated FAQ answering system in the community of its users:

- *Means of information acquisition.* The system's natural language based user interface lets people formulate their problems and submit them as questions to the system that records these questions before answering.
- *Form of organizational memory*. We can perceive organizational memory as storage bins containing information about past decision stimuli and responses [Walsh and Ungson 1991, p. 61]. An FAQ answering system contains:
- Identified problems (stimuli mentioned above). Each FAQ identifies a problem that has appeared within the community of the users of the system.
- Solutions to these problems (responses mentioned above). Each FAQ has an answer that explains the solution to the problem expressed in the FAQ.
- Means of information retrieval. The system retrieves FAQs and their answers upon request expressed in natural human language.

This research is devoted primarily to the last role of an FAQ answering system – means of automated information (i.e., FAQ) retrieval.

Prioritized Keyword Matching

The statistical methods of Information Retrieval [Salton and McGill 1983; Salton 1989] count frequency of common terms in the documents being compared in order to determine similarity between these documents. These methods process large documents and are not appropriate for FAQ answering: single sentences are too

[18] http://ekd.dsv.su.se/faqs.htm, valid in November 2001

short for calculations of term frequency.

On the other hand, semantic language processing like that used by FAQ Finder either requires a very rich lexicon and a knowledge base dealing with the meanings of FAQs (an average website cannot afford such a lexicon and knowledge base) or yields low quality of FAQ retrieval otherwise.

The Prioritized Keyword Matching technique was developed in order to make automated FAQ answering affordable for virtually any website.

Basic Idea

The idea of Prioritized Keyword Matching is based on the assumption that there are three main types of words in a sentence within a certain context in a certain subject:

- *Required keywords* are the words that convey the essence of the sentence. They cannot be ignored.
- *Optional keywords* help to convey the meaning of the sentence but can be omitted without changing the essence of the sentence. The nuances may change though.
- *"Irrelevant" words*, like "a", "the", "is", etc., are words that are too common in ordinary language or in the subject. The meaning of "irrelevant" words is close to that of stop-words in Information Retrieval. The only difference is that stop-words are assumed always unimportant in a given collection of documents, whereas any of the "irrelevant" words in Prioritized Keyword Matching may suddenly become relevant if used in order to emphasis nuances in a particular sentence in a given collection of sentences. The latter happens rarely.

Let us consider an example with "What is the relationship between Business Goal Models and Business Process Models?" In this sentence we distinguish:

- required keywords "relationship", "goal", "process";
- optional keywords "business", "models";
- irrelevant words "what", "is", "the", "between", "and".

If we modify this selection of words with their synonyms and various grammatical forms, we obtain a new, broader selection of words, which characterizes a set of different sentences that are semantically related to the one given above. We assume that these sentences are related although we do not comprehend them.

Let us define that each keyword is always represented by a number of synonyms and their grammatical forms, and that irrelevant words are the same for all the sentences. Hereby, if the same required and optional keywords can characterize two sentences, we declare that both sentences have about the same meaning, i.e., they match each other. This is the basic idea of Prioritized Keyword Matching.

There is also the forth type of words – *forbidden keywords* – whose possible

presence in a sentence is not compatible with the existing meaning of the sentence. For instance, for the sentences "Why do we use it?" and "How do we use it?", "how" and "why" are respectively forbidden keywords: the formulation of the first sentence is not expected to contain "how", the formulation of the second one is not expected to contain "why". In practice, we do not consider all the possible words not expected in the formulation; we consider forbidden keywords only when we need to distinguish two similar sentences having the same required keywords. Forbidden keywords emphasize the difference between both sentences.

One may wonder why "business" and "models" in the example above are optional keywords, i.e., less relevant. The reason is that, in the context of Enterprise Modelling, Business Goal Models and Business Process Models are often referred to as simply goals and processes. A user may formulate the question as follows: "What is the relationship between goals and processes?" "Business" and "models" do not appear in this formulation.

Conceptual Data Structure

Let us assume that we have a database consisting of FAQ entries where each FAQ has its required, optional, and forbidden keywords specified.

According to the basic idea of Prioritized Keyword Matching, each FAQ becomes a pattern that identifies a class of questions with similar meanings, where the keywords of the FAQ identify the concepts relevant to this pattern. After an arbitrary user question is asked the system uses the Prioritized Keyword Matching algorithm to match the question to each FAQ entry separately in order to determine whether or not the question belongs to the class of questions identified by the FAQ.

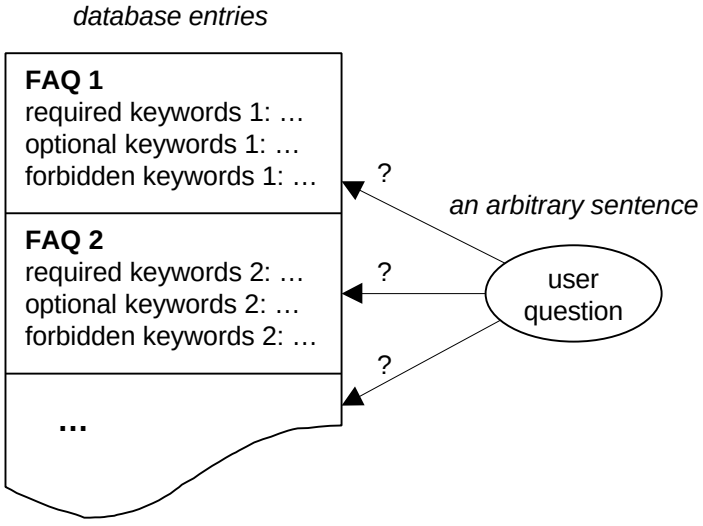

Figure 3 Input to the algorithm: an FAQ entry with identified keywords and an arbitrary user question.

Hereby, the algorithm has the following input (Figure 3 illustrates it):

- an arbitrary sentence – a user question; and
- an FAQ entry with required, optional, and forbidden keywords.

The output of the algorithm is a statement denoting whether or not the user question matches the FAQ in the entry. The algorithm uses a list of "irrelevant" words introduced earlier; there is one such list for all the FAQ entries in the database.

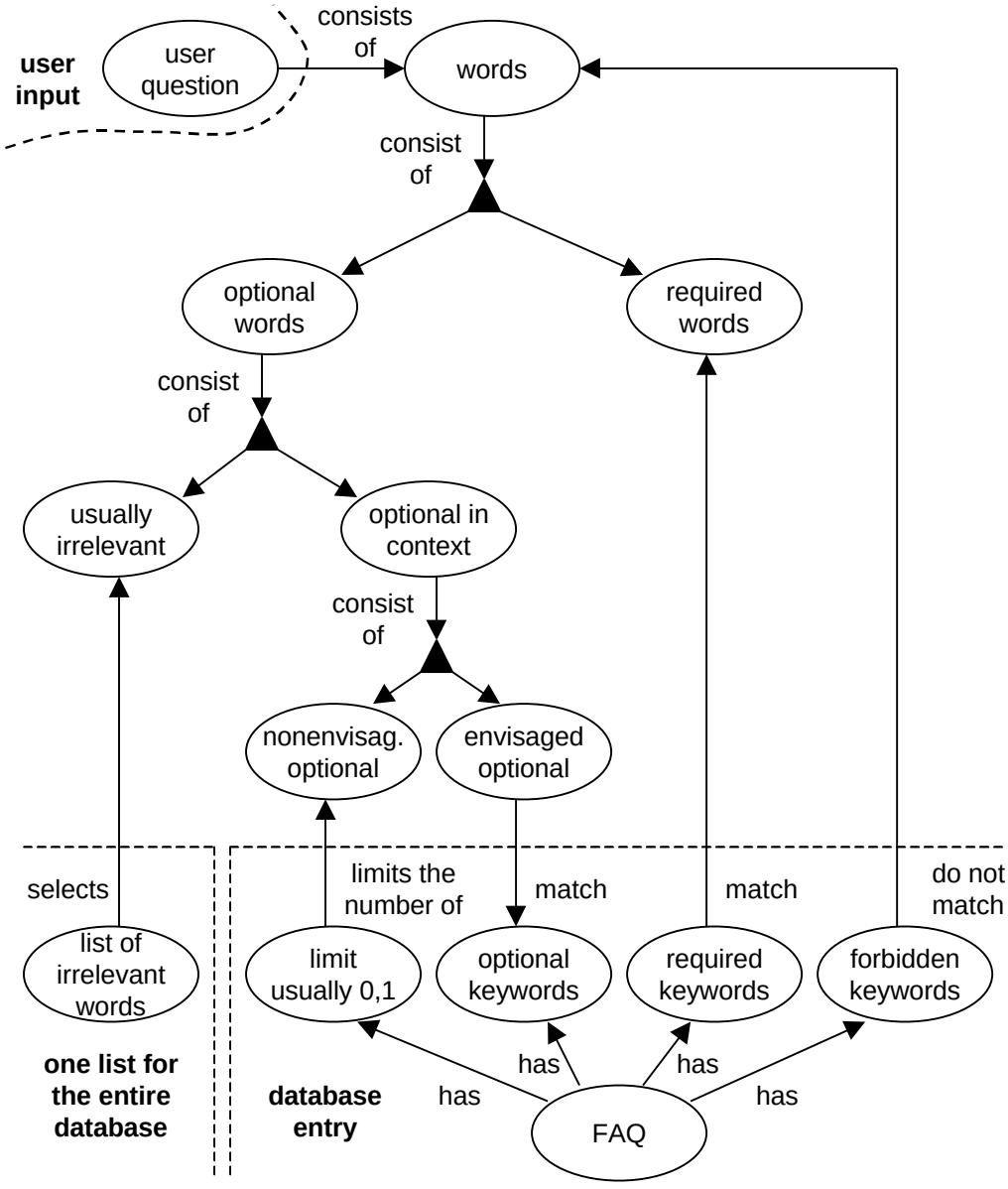

Figure 4 Concepts involved in Prioritized Keyword Matching and the relationships between them if the FAQ answers the user question.

Figure 4 shows the concepts involved in the algorithm and the relationships between these concepts if the FAQ answers the user question. It is important to

note that the *only user's concern is his or her own question*. When typing the question, the user knows nothing about the structure of the database, the keywords, and the matching algorithm. All the data, except the question itself, either come from the database or is created during the matching process. The data concepts are explained in the next subsection along with the algorithm.

Description of the Algorithm

In order to better understand the Prioritized Keyword Matching algorithm, let us observe it together with an example in the context of Enterprise Modelling. After a user has asked a question, the system matches this question to an FAQ entry in the database.

The ***user question***: "How are substantial business goals related to business processes?"

The ***FAQ*** "What is the relationship between Business Goal and Business Process Models?" and its keywords:

- *Required*:
  a) "goal", "goals";
  b) "process", "processes";
  c) "relation", "relations", "relationship", "relationships", "dependence", "dependencies", "connection", "connections", "association", "associations", "link", "links", "linked", "linking", "relate", "relates", "related", "relating", "connect", "connects", "connected", "connecting", "associate", "associates", "associated", "associating".
- *Optional*: "business", "businesses", "model", "models".
- *Forbidden*: none.
- *Limit of non-envisaged words*: 1 (described in Step 6 of the algorithm).

The human common sense says that the user question and FAQ convey roughly the same meaning. The system has to formally determine this by performing the following steps:

1. The system splits the user question into separate words.

   In the example, the question is split into "how", "are", "substantial", "business", "goals", "related", "to", "business", "processes".

2. The system matches the ***required keywords*** in the entry, usually two or three, to the words of the user question. If there is at least one required keyword that is not represented among the words of the user question by at least one synonym or grammatical form, the system *rejects* the match between the user question and the FAQ.

In the example, all three required keywords of the FAQ are represented among the words of the user question: “goals” (a), “processes” (b), and “related” (c).

3. The system matches the ***forbidden keywords*** in the entry, if any, to the words of the user question. If there is at least one forbidden keyword that is represented among the words of the user question by at least one synonym or grammatical form, the system *rejects* the match between the user question and the FAQ.

   In the example, there are no forbidden keywords. These keywords are rarely used only to emphasize the difference between similar in appearance but still different in meaning FAQs.

   After matching the required and forbidden keywords, the system removed their counterparts among the words of the user question and proceeds with the optional words: “how”, “are”, “substantial”, “business”, “to”, “business”.

4. From the optional words, the system filters out those listed as usually irrelevant (“a”, “the”, “is”, etc.). The filtering is based on the ***list of irrelevant words***, one list for all the FAQs in the database.

   In the sample question, irrelevant are the words “how”, “are”, “to”. After they are filtered out, there are only context dependent optional words left: “substantial”, “business”, “business”.

5. The system matches the context dependent optional words of the user question to the ***optional keywords*** in the entry. The system identifies and filters out the context dependent optional words that match these keywords.

   In the sample question, the only context dependent optional word that matches the optional keywords is “business”; in Figure 4 it is referred to as envisaged optional. The other one – “substantial” – does not match the optional keywords; in Figure 4 it is referred to as non-envisaged optional.

6. The system considers the words left – non-envisaged optional words – which match neither required nor optional keywords, and are not in the list of irrelevant words. If there are too many such words, the system *rejects* the match between the user question and the FAQ in the entry. How does the system determine this “too many”? For this purpose, there exists a ***limit*** of non-envisaged words, usually 0 or 1, stated in the entry and dependent on the complexity of the FAQ. The number of non-envisaged optional words may not exceed this limit.

   In the sample question, the only non-envisaged optional word is “substantial”, which does not exceed the limit in this FAQ entry equal to 1. Therefore there is no reason to reject the match between the user question and the FAQ.

7. Already three times the system had an opportunity to reject the match – in Steps 2, 3 and 6. It did not use this opportunity. It *accepts* the match between the user

question and the FAQ in the entry.

Required, optional, and forbidden keywords in an FAQ entry may be represented by both single words and phrases (phrases are discussed further). In order not to corrupt phrases in the user question during the matching process, the words in the question are not removed physically; they are just marked as matching.

A user would loose much information if the system retrieved only FAQs that are very close to the user question. Therefore the system retrieves so called related FAQs as well, as showed in Figure 2. An FAQ is considered related to the user question if all of its required and no forbidden keywords are represented among the words of the question; optional words are ignored. This is checked in Steps 1 through 3 of the above algorithm.

What is a Good FAQ Entry?

There is a simple answer: a good FAQ entry is one which *does* match a large variety of differently formulated user questions with the meaning close to that of the FAQ, and *does not* match not related user questions. Three features characterize a good entry:

- *Thorough selection of required and optional keywords* in an entry highlights representative concepts of the FAQ.
- *Good context dependent controlled vocabulary* (i.e., lexicon) ensures the ability of the system to resolve context dependent synonyms and grammatical forms of each keyword.
- *Sufficient number of auxiliary entries* helps to meet a large number of formulations of a user question. Although the approach of matching required, optional, and forbidden keywords is flexible, sometimes one FAQ entry in the database cannot represent all conceivable formulations of the corresponding user questions. Therefore several entries for one FAQ may be introduced. For instance, “What is Actor and Resource Model?” and “How do we describe actors in Enterprise Modelling?” are two formulations of the same FAQ, each in its own database entry with its own keyword set. In the real system there are 1-2, less often 3 auxiliary entries for each FAQ.

Each FAQ entry in the database has a small lexicon. Synonyms and various grammatical forms of each keyword are considered so that the entry covers as many different ways of asking the same question as possible. Typical grammatical variations are:

- singular and plural forms of nouns;
- tenses of verbs;
- different spellings, American vs. British English (e.g., “model<u>i</u>ng” vs. “model<u>l</u>ing”, “formula<u>s</u>” vs. “formula<u>e</u>”, “analy<u>z</u>e” vs. “analy<u>s</u>e”);

- split and merged words (e.g., “sub-model” vs. “submodel”, “non-existent” vs. “nonexistent” vs. “not existent”).

Typical cases of synonymy are:

- ordinary language synonyms: “related”, “connected”, etc.;
- switching between related verbs, nouns and adjectives: “In what cases do we apply Enterprise Modelling?” vs. “What are the cases of application of Enterprise Modelling?” vs. “When is Enterprise Modelling applicable?”;
- words that are not ordinary language synonyms, but act like synonyms in a particular context: “Why is Enterprise Modelling beneficial?” vs. “Why do we use Enterprise Modelling?”;
- generalization and specialization of a concept (not common).

Prioritized Keyword Matching vs. Techniques of Information Retrieval

The surroundings of the use of Prioritized Keyword Matching resemble those of Information Retrieval: we have a free-text user query and a collection of indexed documents where we perform exhaustive search. In case of Information Retrieval the index means a term vector for every document and a common stop-list; in case of Prioritized Keyword Matching the index means required, optional, forbidden keywords for every document and a common list of “irrelevant” words. Sounds similar.

The principal difference between both techniques is following: importance of a term in a term-vector is denoted by its scalar weight whereas importance of a term in case of Prioritized Keyword Matching is denoted by its *non-scalar* role (i.e., required, optional, etc. keyword). Knowing the role of a term in the document we make much better conclusions about different properties and the importance of the term than just knowing the weight as the only property. This core difference has the following consequences:

- Term-vectors are effective only if they are long enough unless there is additional information, other than scalar proportion of the importance of the term, encoded in the numerical weight. On the contrary, the roles assigned to the terms in a document do not require many terms in the document in order to compare it to another document.
- In Prioritized Keyword Matching, the roles are assigned to the terms in the collection of documents only; the user query is not indexed. On the contrary, the term-vectors in Information Retrieval require indexing of both the query and the documents in the collection.

The major drawback of roles is that we need intelligent reasoning in order to assign a role to a term. On the contrary, in Information Retrieval we assign weights to the terms according to the corresponding term frequency with no reasoning

whatsoever.

Idea of Multiple Lexicon

FAQ entries in the system's database, once created or updated, are static. The keywords of each FAQ are known long before any user questions are asked. Therefore the synonyms and grammatical forms of each keyword are put into the entry along with the keyword. No external source of lexical information is used during the matching of a user question to this entry. Lexical and morphological analysis of the keywords in the entry is done before these keywords are used.

Multiple vs. Single Lexicons

One may suggest that the system uses no lexicon. This is not true. Each FAQ entry has a small lexicon that implements one function – identification of mutually exchangeable words (synonyms and their grammatical forms) for every keyword within the context of a given FAQ. The system uses a multiple lexicon assembled from numerous independent small lexicons, where each of them is attached to its own FAQ entry. Figure 5 illustrates the difference between multiple and the ordinary single lexicons.

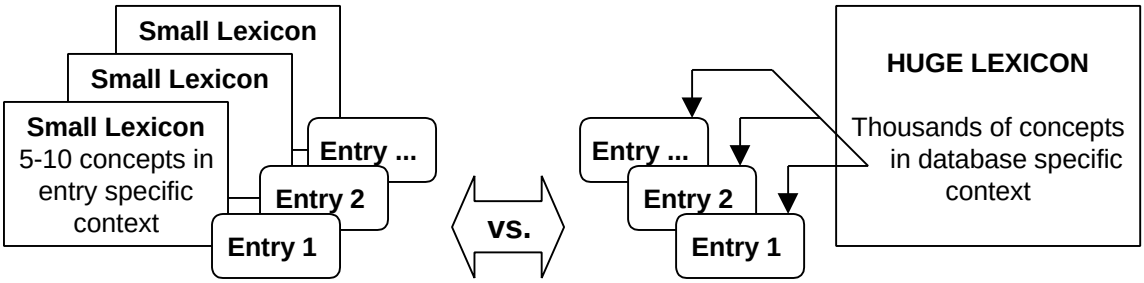

Figure 5 Multiple (many small units) vs. single (one large unit) lexicons.

The principal difference between both types of lexicons lies in the size and context of autonomous units. A single lexicon has one large unit containing thousands of concepts within the context of the entire database, whereas a multiple lexicon has many small autonomous units containing 5-10, or even less, concepts within the context of one database entry (i.e., one FAQ). A multiple lexicon has certain advantages:

- Semantic relationships between words may be entry rather than database specific, like in "How do we develop a Business Process Model?" vs. "How do we acquire a Business Process Model?". It is not possible to include so specific relationships in a lexicon one for the entire database.
- It is difficult to maintain a large lexicon. As new concepts are entered into such a lexicon they get "frozen". If a concept or its relationships with other concepts are modified, we must keep track of how the changes interact with the context

of every single entry where this concept is used. The units of a multiple lexicon are more localized, therefore their maintenance is less error prone.

- Traditionally in natural language processing, grammatical analysis is applied to a user query. A lexicon, traditionally the single one, is a tool of this analysis. A multiple lexicon offers another model of the analysis: in case of one-sentence database entries like in FAQ answering, an autonomous unit of the lexicon stores the result of pre-made analysis of the sentence (i.e., FAQ) rather than a tool for this analysis. Hence:
  - we can use whatever advanced language processing tool we want (because we store the result of the analysis); and
  - the query processing is reduced to keyword matching, which requires simpler data structure and less processing power than analysis of the query using a single lexicon.

A possible drawback of a multiple lexicon is redundancy. Yet this is a minor drawback. If the database has 300 entries with 10 concepts each, there are only 3000 concepts in total, which is less than a small dictionary anyway. The target system of this research reduces redundancy by using substitutes (substitutes are discussed further).

The idea of multiple lexicon is analogous to that of object-oriented programming (OOP):

- *Definition of a lexicon* (in OOP – definition of a class). The necessary functions of lexical analysis – resolving of synonyms, grammatical forms, generalization, specialization, etc. – are defined (in OOP – definition of methods) using empty concept slots (in OOP – properties or attributes) since the actual concepts are not known yet at the definition stage.
- *Instances of the lexicon* (in OOP – instances of the class, i.e., objects). Within a series of different contexts, the concept slots created during the definition phase are filled by the actual concepts. Now there is a series of analogous, autonomous, narrow context dependent lexicons where each lexicon may contain five concepts as easy as five thousand.

Role of Human Reasoning in FAQ Answering Using a Multiple Lexicon

The Prioritized Keyword Matching technique, which uses a multiple lexicon, was developed for an FAQ answering system considering the following peculiarities:

- The task of the system is automated FAQ answering; the system itself does not introduce new FAQs.
- Human intelligence is easier to “implement” then artificial intelligence.

- Since FAQs about the topic (particularly, Enterprise Modelling) were not collected previously, the FAQs must be written and the database must be populated manually by the administrator (not a user!) of the system. Therefore we can use full advantage of the present human reasoning and ask the administrator to select the keywords and create autonomous units of the multiple lexicon as well. The administrator does enjoy computer assistance during this work, but this is a subject of other research, not that of FAQ retrieval.

We can imagine FAQ entries as pieces of conserved human intelligence; pre-made decisions are applied upon a user's request as a query is submitted, as opposed to artificial intelligence where decisions are made upon a user's request. Hence, using a multiple lexicon we can reduce the task of FAQ retrieval to keyword matching and still have an illusion of an intelligent system.

Well, FAQ entries for the target system of this research are created manually. Yet this work is neither too tedious nor difficult. The entries are reusable and copy-and-paste manipulations applicable. Synonym groups in the context of different FAQs are not always the same but usually similar. Computerized tools – dictionaries, grammar prompters, spelling checkers, you name it, incorporated in the administration tool or as stand-alone applications – are possible and welcome as long as the human supervision is preserved. The result of the efforts – an FAQ entry – is important whatever methods are used in order to create it.

The approach to FAQ answering using a multiple lexicon was designed for a particular system operated under particular circumstances – the system maintains its own FAQ set. Therefore the approach should *not* be misleadingly generalized; it should *not* be misapplied to systems built for a different purpose – navigation through an external frequently changing FAQ source.

Regarding involved human resources, the Prioritized Keyword Matching technique was developed, the FAQ answering system designed and implemented, a particular FAQ set created and maintained – this work was done by one postgraduate student.

Substitutes

Let us come back from a theoretical discourse to more practical issues. Since the Prioritized Keyword Matching technique performs formal keyword matching without understanding the meanings of the words, we can introduce a shortcut for a group of context dependent synonyms and their grammatical forms with similar appearance. For instance, "relat*" can be a shortcut for "relation relations relationship relationships relate relates related relating". The only meaning of the shortcut is a graphical substitute for a group of words. While shortcuts are not visible to the users of an FAQ answering system, they make administration of the system easier. With shortcuts, the sample FAQ entry discussed along with the

Prioritized Keyword Matching algorithm looks more attractive. The FAQ: “What is the relationship between Business Goal and Business Process Models?” The keywords:

- *Required*:
  a) “goal*”;
  b) “process*”;
  c) “relat* depend* connect* associat* link*”.
- *Optional*: “business* model models”.

The optional keyword “model” has no shortcut in order to distinguish it from “modelling”.

One may object that “goal*” matches both “goal” and “goalkeeper”. It is not likely, however, that the system maintaining the above FAQ could get a question where soccer players and processes along with their relationships would be combined into one sentence within the context of Enterprise Modelling. While shortcuts make the work of the administrator easier, they are not enforced where they are not appropriate.

We may observe that, although synonym groups differ from context to context, they may have common, repeating words. In order to save writing efforts, we can create a repository of substitutes for repeating groups of words. For instance, we can define “$models” as a substitute for “model models”, put it into the repository of substitutes, and use like this:

- *Optional keywords*: “business* $models”.

Here “$models” has no other meaning as a graphical substitute for the two words. There can be shortcuts used in the definition of a substitute.

Existence of a repository of substitutes does contradict with the idea of multiple lexicon because the units of the lexicon stop being autonomous – they have common substitutes. Nonetheless, the advantages of a multiple lexicon are preserved if substitutes are used carefully. Substitutes save writing efforts, and it is up to the administrator of the system to decide where and how to use them.

Phrases

The first version of the FAQ answering system developed within the scope of this research did not recognize phrases; it did not distinguish “process modelling” from “modelling process”, which was an obvious disadvantage to be eliminated.

What is a Phrase for Prioritized Keyword Matching?

A phrase in a user question is a sequence of words where their order is important. A phrase represented in an FAQ entry is a sequence of concepts where each concept

is represented by a group of synonyms and their grammatical forms. Each synonym may be a single word or another, embedded phrase. The administrator of the system enters a phrase into an FAQ entry along with the keywords as one of the synonyms of a keyword according to the following syntax: "<" denotes the beginning if a phrase, ">" denotes the end of a phrase; ";", ":", and "#" are delimiters between the concepts in the phrase. Examples:

- <process*; modelling modeling>
- <<modelling modeling; process*> # <in; spite; of> despite # <process*; modelling modeling>>

There are three types of concepts in a phrase:

- "<" and ";" are delimiters in front of a mandatory concept: "<one; of; two three>" matches either "one of two" or "one of three" and nothing else.
- ":" is a delimiter in front of an optional concept: "<on: the; other; hand>" matches "on the other hand" and "on other hand" with dropped "the".
- "#" is a delimiter in front of a concept that allows having any number of any words between this and the previous concept: "<modelling modeling # process*>" matches both "modeling process" and "modelling of many different kinds of various processes" (note that both have different meanings).

A user of the system does not see how phrases are represented in an FAQ entry.

Main Ideas behind the Phrase Processing

The reasoning in this subsection is not even of the concern of the administrator of an FAQ answering system; the subsection discusses the principles of matching a phrase to a user question implemented in the target system of this research.

During the matching process, the system constructs a graph for each phrase in an FAQ entry. Matching of a phrase starts when the first concept in the graph matches some expression – a single word or another, smaller phrase – in the question. Supposedly, the rest of the concepts in the graph should match the rest of the expressions (mostly single words) in the question. Nonetheless, the matching is not straightforward because concepts may be optional, there may be variable distance between adjacent concepts, or several synonyms (which may be embedded phrases of different length, and so on recursively) in a concept can match an expression (not necessarily the same) in the user question. If we can match the same phrase graph in many different ways and get different results, we have alternative paths in the control flow of the matching. It is possible to construct a representation of a phrase so that no alternative paths ever appear; a reliable system, however, must be able to process them in case if they do appear. In order to make it possible, the control flow in the phrase graph must be organized properly. Figure 6 shows incorrectly and correctly organized control flows.

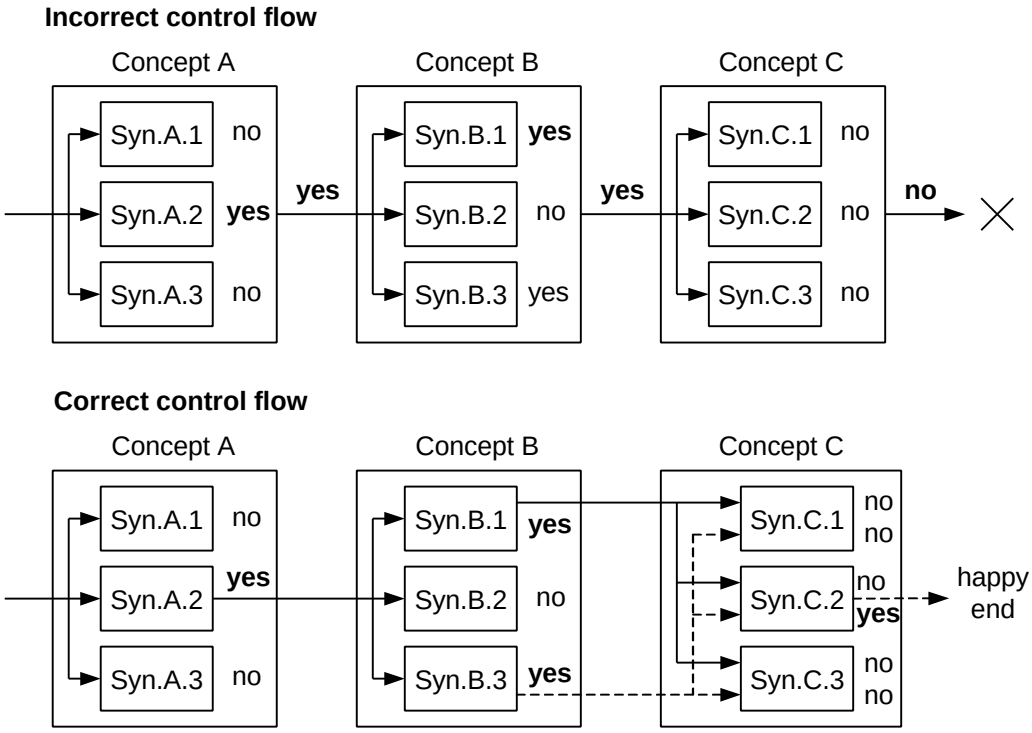

Figure 6 Incorrect and correct control flows of matching a phrase.

An incorrect control flow goes from concept to concept: the system discovers that there is a synonym in a concept that matches an appropriate expression in the user question and proceeds with matching the next concept. Concept B, however, turned out a trap: there were two matching synonyms. The system took the first one – Syn.B.1 – and failed at Concept C. It was too late to return to Concept B and try the alternative Syn.B.3 because the information about previous alternatives was already lost.

A correct control flow goes from synonym to concept. After the system had selected Syn.B.1 and failed at Concept C, it came back to the "fork" in Concept B and took Syn.B.3, proceeded with Concept C one more time (the dashed line), and reached the happy end.

A correct phrase graph should be constructed so that there is a link from each synonym of the current concept to the next concept. If the synonym is a single word, the link goes from this synonym to the next concept. If the synonym is an embedded phrase, the link goes from each synonym of the last concept of the embedded phrase to the next concept in "this" phrase, and so on recursively.

Preliminary Evaluation of an Implementation of Prioritized Keyword Matching

Two core features influence the performance of an FAQ answering system: quality of the language processing (as a question is asked, the system must find the

corresponding FAQ) and completeness of the FAQ set (the corresponding FAQ must exist in the database).

A discussion on completeness of the FAQ set is presented in [Sneiders 1999-a]. The main issue we should consider here is following. In the beginning, the system is able to answer only a few questions. We may ask a number of questions and do not get them answered, which indicates that the system has a bad question answering ability. After these questions are incorporated into the FAQ set, which is normally done, we may ask them once more and get them answered, which indicates that the system has a perfect ability to answer questions. The universe of conceivable questions of any reasonably broad topic is infinite, completeness of the FAQ set is ever improving, the system is evolving.

The quality of Prioritized Keyword Matching technique is determined by its ability to retrieve relevant existing FAQs from the database upon a user's request. In Information Retrieval there are two parameters to measure the quality of such retrieval – recall and precision [Salton and McGill 1983, pp. 164-172; Salton 1989, pp. 248-249, 277-278]. *Recall* R characterizes the system's ability to retrieve all the relevant items existing in the collection of documents (i.e., FAQs):

$$R = \frac{\text{number of relevant documents retrieved}}{\text{total number of relevant documents in collection}}$$

Precision P characterizes the system's ability to retrieve only relevant items:

$$P = \frac{\text{number of relevant documents retrieved}}{\text{total number of documents retrieved}}$$

Although all the features of language processing discussed in this paper are implemented and tested, there is not enough empirical data for a formal evaluation. Instead, recall and precision of the first implementation of Prioritized Keyword Matching is presented. This implementation has no phrase processing, no forbidden keywords, no shortcuts, no substitutes. At the moment of taking the measurements, there were more than 80 questions asked to the system (asked by people other than the administrator of the system) and logged. From those more than 80 recorded question / reply pairs, selected were those where the questions were not duplicate (which happened if questions were posted for demonstration purpose) and without spelling mistakes. In order not to be influenced by completeness of the FAQ set, only those question / reply pairs were considered where the question had the corresponding FAQs in the database at the moment of asking it. Eventually, not so many – 17 – question / reply pairs were selected that satisfied the criteria above.

At first, only close answers were observed ignoring related FAQs. The average recall was 0.65; the average precision was 0.95. Nonetheless, it proved that often the system incorrectly classified a close answer as a related FAQ. Since the user obtained the FAQ anyway, recall and precision ignoring the difference between

close answers and related FAQs was worth measuring: the average recall was 0.88, the average precision was 0.85. These figures are high. Although the measurements with only 17 queries may be criticized for not being representative, they shed positive light on the potential of the technique.

Prioritized Keyword Matching showed good query processing time – 1 to 2 milliseconds to match a user question to a separate FAQ entry (roughly 0.5 milliseconds more after phrase processing was introduced). The processing time is subjective: it depends on the hardware, efficiency of the compiler, skills of the programmer. The particular FAQ answering system was operated in the Microsoft Windows NT 4.0 environment using a Pentium 166 MHz processor and 48 MB RAM.

An experienced administrator of the system needs 5-15 minutes in order to select and test the keywords for an FAQ entry.

Further Research and Conclusions

There are a number of possible directions of developing the ideas implemented in the target system of this research. Several of them are mentioned below.

Although the administrator of the system enjoys computer support when he or she creates an FAQ entry, an integrated tool-set would ease the tasks of selecting and analyzing the keywords. Additional support is needed in order to create the initial set of FAQs before the system is put into operation since no one is going to ask any questions to a system with no FAQs. Automated analysis of manuals and similar literature could suggest raw material for the empty database of a newly created FAQ answering system.

The Prioritized Keyword Matching technique uses a multiple lexicon. It would be a challenge to move forward into deeper language analysis using this kind of lexicon. We could change the categorization of required, optional, and forbidden keywords so that it is stated who does what. Then the system could better suggest related FAQs and give better hints on what kind of related information exists in the database. Combining this with parsing of user queries, the system may try to resolve pronoun references “it”, “this”, “that”, etc., and make kind of alive dialogue with the user.

According to the basic idea of Prioritized Keyword Matching, an FAQ in the database is a pattern that identifies a class of questions with similar meanings. Required and optional keywords identify the concepts relevant to this pattern. Each keyword is represented by a group of synonyms. In FAQ answering, the pattern is a human language sentence, the keywords identify concepts expressed in human language, and the synonyms are natural language words. We may generalize the pattern mechanism of Prioritized Keyword Matching and apply it to social, physical, chemical, etc. phenomena. We can organize generic patterns of these

phenomena like FAQs in a database and search through the database by using the Prioritized Keyword Matching technique.

Conclusions

This paper presents continued research in automated FAQ answering by using shallow language understanding. The Prioritized Keyword Matching technique discussed here was developed in order to match an arbitrary user question to an FAQ entry in the database. The use of shallow language understanding means that the matching is based on keyword comparison; the system performs no syntactic parsing of the question, it does not extract semantic concepts. In Prioritized Keyword Matching, lexical and morphological analysis is applied to the keywords in the FAQ entries rather than user questions long before any questions are submitted. This implies use of a multiple lexicon assembled from numerous autonomous FAQ-context dependent small lexicons: each of those small lexicons is attached to its own FAQ entry. We can imagine FAQ entries as pieces of conserved human intelligence; pre-made decisions are applied during the matching process as a user query is submitted. Hence, by using a multiple lexicon we can reduce the task of FAQ retrieval to keyword matching. A system using Prioritized Keyword Matching may attain good recall and precision of FAQ answering. Two earlier obtained (recall, precision) value pairs are (0.65, 0.95) and (0.88, 0.85). The lion's share of this success is attributed to the context dependence of the autonomous units of a multiple lexicon.

The approach to FAQ answering using a multiple lexicon was designed for a particular system operated under particular circumstances – the system maintains its own FAQ set. Therefore the approach should *not* be misleadingly generalized; it should *not* be misapplied to systems built for a different purpose – navigation through an external frequently changing FAQ source.

The paper introduces an original approach to processing phrases within the framework of shallow language understanding. A phrase is recursively represented as a series of concepts, where each concept contains synonyms, where each synonym may be a single word or another, embedded phrase. While matching a phrase, the system is able to process alternative paths.

Relative simplicity of the Prioritized Keyword Matching is aimed at making automated FAQ answering affordable for an average website. By having a natural language based user interface, the system adds one more dimension – limited human language understanding – to the traditional notion of multi-media technology (images, sounds, animation) on WWW. One person with at least normal intelligence is able to install the software and populate the FAQ set of the system. The system is ready to work with the first FAQ entry in the database; neither a large lexicon nor a knowledge base for inference and deduction are needed.

There exists a version of the working system which answers questions on Enterprise Modelling. Another version, which answers questions on Internet protocols, is being introduced.

Application and Maintenance Aspects of an FAQ Answering System

Introduction

The amount of information available on an average computer grows rapidly. Information management and communication technologies try to help people feel comfortable in the flood of this information, which in its turn rises the risk of getting lost in the jungle of both the information itself and the associated technologies. People traversing the information space either easily find what they are looking for or give up.

Supposedly, the most natural kind of queries posted to an information system is questions stated in ordinary human language. Unfortunately the problem of understanding natural language queries is far from being solved. Nonetheless, FAQ lists – static compilations of typically asked questions and their answers – on the Internet brought up an idea of automated FAQ answering, a simpler task of matching a user question to pre-made FAQs in the system's database without in-depth understanding of the query itself. Automated FAQ answering reduces two shortcomings of existing FAQ lists. First, the users of such lists do not ask any questions, therefore the information provider does not know all the variety of questions that arise. Not many people use e-mail feedback. Second, finding valuable information in a long – several hundreds of entries – and chaotic FAQ list, or a number of lists, is a tedious work.

The article presents a WWW (World Wide Web) based FAQ answering system. The system is referred to as FAQ AS (i.e., FAQ Answering System) in order to distinguish it from similar systems. FAQ AS has been tested using two FAQ collections, one on Enterprise Modeling[19] and the other on HTML[20]. In order to illustrate the system's functionality, an example of asking a question follows. By using a web browser, the user submits his or her question to the system (Figure 1).

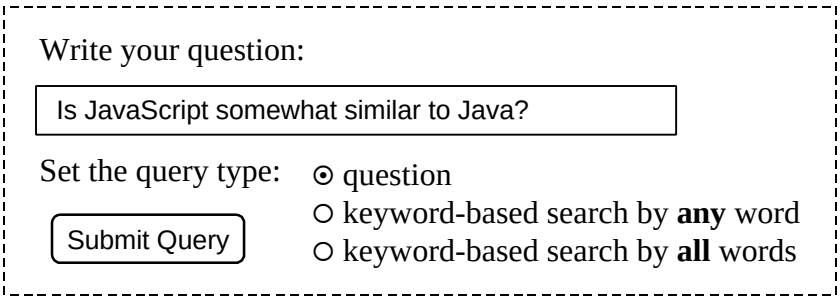

Figure 1 Example of a user question.

[19] http://ekd.dsv.su.se/faqs.htm, valid in November 2001

[20] http://www.dsv.su.se/html/, valid in November 2001

The system receives the user question, logs it, and searches through the database in order to find FAQs that correspond to the question. After one or several, if any, relevant FAQs are found, the system returns them and their answers to the user, as shown in Figure 2.

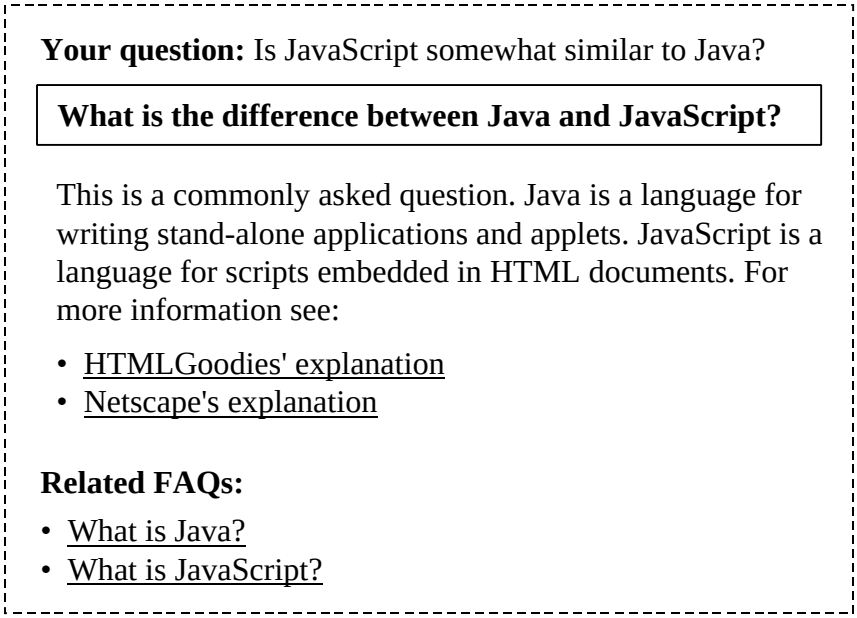

Figure 2 Reply to the question.

A more common alternative to answering natural language questions is keyword-based search used by most search engines. Many people, however, do not think in terms of Boolean expressions made of keywords and are not used to the engine-specific syntax of such expressions. As a rule, keyword queries retrieve a large amount of irrelevant information. After all, it is more fun to talk to a computer in ordinary English. A natural language based interface does indirect interviewing of the users: in the logs of the system we can read what people think when they search for information. On the web such interface adds one more dimension – limited human language understanding – to the traditional notion of multi-media (images, sound, animation).

A number of natural language question-answering systems have been developed until the middle of 1990-ies, such as those in [Grosz et al. 1986]. None of them, however, became well known because question-answering systems are complex and somewhat useless if they serve only a limited number of experts in the question domain. The situation has changed as the development of the Internet has made information systems accessible to a huge laymen audience which wants to enjoy a simple interface. As a popular means of communication, the Internet has given a new force to research in automated question answering.

The first Internet related FAQ answering systems appeared in the middle of 1990-ies. FAQ Finder [Hammond et al. 1995; Burke at al. 1997-a] is in some sense a

browser aimed at improving navigation through the numerous FAQ files in Usenet newsgroups. Upon a user's natural language question, the system located relevant FAQ files and retrieved relevant question-answer pairs from them.

Another system – Auto-FAQ [Whitehead 1995] – was powered by the idea of joining small efforts of many people in order to accumulate information, which became possible thanks to the system's web-based interface. Auto-FAQ answered user questions from its own infobase. If a question could not be answered, it was placed into a public "gap list". The users who were experts in the particular domain could answer the question and move the newly created question-answer pair to the infobase.

The Winiwarter's natural language interface [Winiwarter 1999] used the question-answering solutions of Artificial Intelligence (AI). The system used machine learning to learn questions from FAQ collections. Similarly, it used machine learning to learn user questions and match them to the FAQs. Although this process requires continuous user input, a positive aspect is that it accumulates small efforts of many users.

The article has twofold aims. First of all it discusses where and why to use an FAQ answering system. FAQ AS is used as an example of an FAQ answering system. Second, it presents the administration and maintenance aspects of FAQ AS as an example of what it can take to run an FAQ answering system. The article does not focus on the language processing technology which is published in [Sneiders 1999-b].

The rest of the article is organized as follows. After a brief overview of the main principles of FAQ AS is given, the user interface of the system is presented. The section on applications of the system discusses incentives and prerequisites of the use of FAQ AS, and a few sample application domains. The discussions on the user interface and applications are generic, they may pertain FAQ answering systems that use different FAQ retrieval technologies. The section on administration and maintenance of the FAQ database, however, is specific to FAQ AS. Finally, the effectiveness of FAQ answering using FAQ AS is briefly evaluated.

Main Principles of FAQ AS

The discussed experiences in automated FAQ answering were obtained along with the development of FAQ AS. The main principles of this system are presented below.

Template-Based Question Answering

Unlike traditional AI question-answering systems that generate new answers, FAQ answering systems retrieve pre-made question-answer pairs from their databases. The knowledge domain of a system is covered by a number of FAQs (Figure 3). Each FAQ embodies a problem statement, and the FAQ answer presents a solution to the problem. As small pieces of information, they are like patches on the knowledge domain.

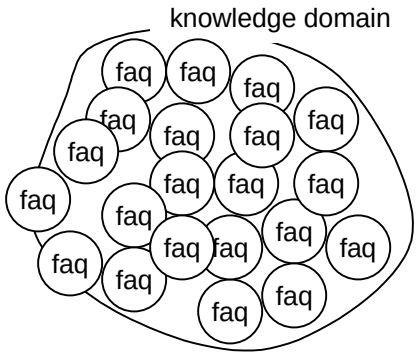

Figure 3
FAQs covering a knowledge domain.

Each FAQ entry in the system's database is a template which represents a class of similar questions. In order to answer a user question, the system matches it to the FAQ entries (i.e., question templates) and finds the FAQs, if any, close enough to the question.

Prioritized Keyword Matching

Let us have a brief look at the Prioritized Keyword Matching technique [Sneiders 1999-b] used by FAQ AS to match a user question to FAQ entries in the system's database. Each entry contains a number of keywords that distinguish its FAQ. Every keyword is represented by several synonyms where a synonym is either a single word or a phrase. There are three types of keywords in each entry:

- *Required keywords* are words that convey the essence of the FAQ. The FAQ and user question are considered close enough only if all the required keywords are represented in the user question.
- *Optional keywords* help to convey the meaning of the FAQ but can be omitted without changing its essence. The nuances may change though. By matching both required and optional keywords, the system makes subtler conclusion about the closeness between the FAQ and the user question.
- *Forbidden keywords* (or negative keywords) denote concepts that are not compatible with the meaning of the FAQ. They help to emphasize the difference between two similar in appearance but different in meaning FAQs. If a forbidden keyword is represented in the user question, the match between the FAQ and the question is rejected.

One FAQ may be represented by several question templates – auxiliary FAQ entries – which consider alternative forms of the question. An alternative form has a different combination of relevant concepts and, therefore, a different selection of keywords.

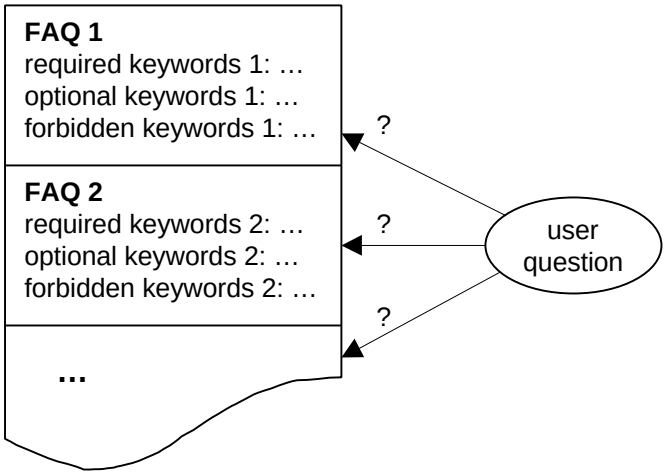

Figure 4 Matching of a user question to FAQ entries.

When trying to answer a user question, the system matches the question to the keywords in every FAQ entry (Figure 4).

FAQ entries in the system's database, once created or updated, are static. The keywords in an entry are selected by doing conceptual, lexical, and morphological analysis of the FAQ when it is entered into the database. The analysis is done by the administrator of the system applying "manual" *human reasoning*. Certain routines are automated, as it will be discussed further. Still, it is important to note that when typing a question, the user knows nothing about the data structure, keywords, and matching algorithm.

Original, Persistent, and Evolving FAQ Collection

FAQ AS was designed for question answering from an original and persistent FAQ collection: an FAQ entry contains unique information and, once created, remains in the collection.

The FAQ collection evolves as more questions are asked, recorded, analyzed, new FAQ entries in the database created and existing ones modified. Not only does the FAQ collection grow; each FAQ entry evolves too. If an entry fails to match a corresponding user question, the entry is updated so that next time it does match this and similar questions. Well-drilled entries accumulate past experiences, therefore their ability to match appropriate and ignore inappropriate user questions increases.

Creating of FAQ entries for Prioritized Keyword Matching requires intelligent reasoning. At the current stage of development FAQ AS cannot perform quick indexing of external data, therefore it cannot be used as a navigation tool through frequently changing information sources.

A Similar Approach

Ask Jeeves[21] is the first commercially successful large-scale question-answering system with an ambition to answer any question. Apparently Ask Jeeves strives to become a natural language based entry to the Internet – a smart alternative to existing keyword-based search engines. According to personal communication with knowledgeable people and bits of information on the Ask Jeeves website, FAQ AS and Ask Jeeves use similar question-answering techniques, although they are developed independently. "Ask Jeeves uses proprietary natural language technology to match your question to one of thousands of 'question templates' and millions of researched links in its knowledgebase. The Ask Jeeves knowledgebase is built by humans who understand what people need when they have questions. Ask Jeeves is continually expanding and becoming smarter over time as people ask more questions"[22].

FAQ AS Architecture

Figure 5 shows the main software components of FAQ AS.

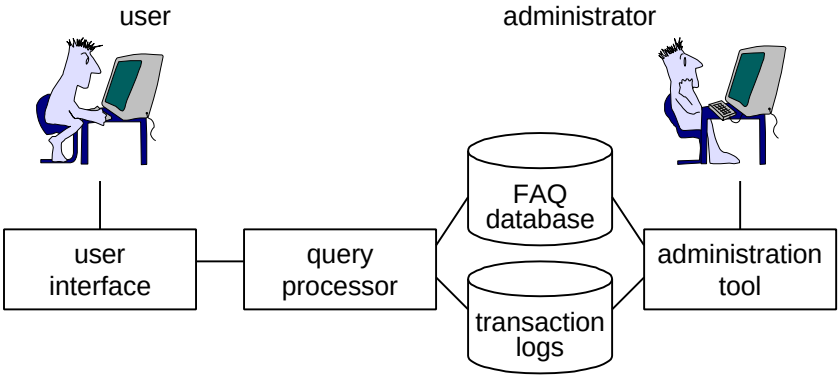

Figure 5 Architecture of the FAQ answering system.

Any web browser that supports HTML forms can be a user interface tool. A user submits his or her question to the query processor which tries to answer it from the FAQ database. The question and the corresponding system's reply are recorded in the transaction logs. By reading the logs, the system's administrator can find out the asked questions, evaluate the system's performance, and make the necessary adjustments in the FAQ database. Thus asynchronous communication between the users of the system and its administrator is established.

[21] http://www.askjeeves.com/, valid in November 2001

[22] The source – a page on www.askjeeves.com – is not available any longer

User Interface

The user input of FAQ AS is implemented by a static HTML document. A user writes his or her question and submits it by pressing the button as shown in Figure 1. This is basically it.

The full range of the system's functionality is broader than just question answering. A user can also do keyword-based search through the FAQ database. The system matches user keywords to the required keywords in each FAQ entry. Because every required keyword is represented by a number of synonyms, concept matching rather than substring search is performed.

On the input page, it is recommended to emphasize the distinction between both types of queries – a natural language question and individual keywords. The initial prompt in front of the input field "Write your question / keywords" was confusing because people were used to keyword-based search engines. After the word "keywords" was removed from the prompt, the proportion of questions among all queries went up remarkably. On the other hand, instead of what-is-this type of questions people often write only distinguishing keywords. For instance, "css" instead of "What is Cascading Style Sheets?"

It happens that by mistake people submit a question to keyword-based search. The response to such a query is a long puzzling list of FAQs not related to the asked question. In order to avoid this, the input form has an embedded JavaScript procedure which checks whether or not user keywords contain the words typical for questions: "is are was were be been how where what when which who why". If such a keyword is discovered, the user is alerted.

Figure 1 does not show the input field for the user's e-mail address. This is optional information useful when the system fails to answer the question. Then the system's administrator can contact the user by e-mail.

The system may let a user list all the FAQs in the collection by simply clicking on a hyperlink. The list items are grouped by their common focus. Each item is further hyperlinked to the corresponding question-answer pair in the database. It was observed that many users found it more convenient to click and list a small number of FAQs rather than to explicitly type a question. Supposedly people prefer typing a question to scanning a list if the number of FAQs in the list is many hundreds.

The reply to a question, which is a dynamically generated HTML document, contains the user question itself and one of the following: (i) one or two, rarely more, *close FAQs* and their answers that most probably answer the question, (ii) *related FAQs* that do not answer the question but explain certain concepts in it, (iii) a combination of both as shown in Figure 2, or (iv) a "not found" message.

The code of the FAQ answer contains HTML tags, which means that in the answer we can use links to other web documents, images, JavaScript, etc.

If a user question contains words that have no counterparts in the FAQ database (i.e., no synonyms of any required, optional, or forbidden keyword match such a word), these words are printed out so that the user can stop looking for concepts that are known as missing in the database. For instance, if a user asks “Is Java in Japan the same as in Europe?”, the system retrieves the related FAQ “What is Java?” (note that this is not the answer to the question) and informs that the words “Japan” and “Europe” were not recognized.

If a user question is answered by close FAQs, the system provides a simple evaluation form where the user can rate how well the answer corresponds to the original question. The grades range from “1” (does not correspond) to “5” (does correspond).

All user queries and brief information about the system’s replies are stored in the transaction logs. If a query has not been satisfied, the system immediately notifies the administrator with an appropriate e-mail message.

FAQ AS Application Domains

FAQ AS plays several roles in the community of its users:

- *Means of information acquisition.* A natural language based user interface lets people formulate and submit their problems as questions to the system that answers and records these questions. The system collects explicit problem statements which could be difficult to elicit by other means.
- *Form of organizational memory.* [Walsh and Ungson 1991, p. 61] refer to organizational memory as storage bins containing information about past decision stimuli and responses. We can relate FAQ entries to these storage bins: each entry contains (i) a question as an identified problem (i.e., a decision stimulus) and (ii) the answer to the question as a solution to the problem (i.e., the response to the stimulus).
- *Means of information dissemination.* The system retrieves FAQs and their answers from its database upon requests expressed in natural human language.

The Answer Garden system [Ackerman and Malone 1990; Ackerman 1994] was a tool designed to help improve an organization’s memory by providing a database of answers to commonly asked questions. Answer Garden was, in fact, an FAQ answering system, a predecessor of FAQ AS. Not having a natural language based interface, the system provided a branching network of diagnostic questions that lead to the answer in the end of the path. What were the incentives for questioners to use Answer Garden? Ackerman argues that people who seek information prefer the channels of the highest accessibility to the channels of the highest quality; the latter usually are busy human experts. Also, contacts with human experts give a possibility of being seen as incompetent. What were the incentives for experts and

organizations? Answer Garden provided faster, easier answers. It reduced the need to have a large amount of experts' time devoted to repeatedly answering the same simple questions over and over again. It allowed codifying and storing electronically more of the knowledge that was stored informally in the minds of people.

Experience with FAQ AS has shown an important, possibly underestimated requirement for successful operation of a question answering system. It is the *target audience*. The system must have a well-defined audience of active users interested in the topic the system covers. A system "to whom it may concern" will probably concern no one. It is wrong to assume that the system will be used just because it exists. At the early stage of this research, an FAQ AS prototype turned out lacking users because there were too few people interested in the information the system rendered.

The audience must be large in order to motivate the efforts of development and maintenance of a natural language based interface. A question answering system set up for a small group of users will fail unless it is a research prototype, or the users enjoy VIP status. Hence, the system must be easily accessible by means of widespread communication technologies like WWW, e-mail, SMS (Short Message Service) or WAP (Wireless Application Protocol) for cell phones, ordinary telephone if the system has voice recognition, or any proprietary communication technology in a large organization.

What is an Appropriate Application Domain?

A typical domain for an FAQ answering system has a large number of complex definitions. The domain comprises a large number of concepts and their interactions. The concepts have very few instances. A number of rules define behavior of the concepts; there are many exceptions to the rules. Two examples of a typical domain are technology and governmental institutions, such as tax and immigration authorities.

Domains not appropriate for an FAQ answering system have a large number of data instances typically stored in a relational database. Automated FAQ answering is not rewarding in the domains that lack complex definitions or have too few concepts. Two examples of an inappropriate domain are a timetable and a warehouse.

Let us distinguish a few feasible application areas of FAQ AS.

Knowledge Repository in an Organization

The idea of building FAQ AS – a web-based system which answers natural language questions – appeared within a research project which dealt with application of Enterprise Modeling in solving organizational problems in two power supply companies [ELEKTRA 1996]. The project participants came from academic and business environments in Greece, Israel, France, Sweden, and UK. In

the beginning they lacked a common understanding of what Enterprise Modeling was and how it should be applied. Despite the differences, these people had to join their efforts in achieving a common goal.

According to the intention of the developers of the applied Enterprise Modeling technique, the FAQ database would become a part of the knowledge repository for the technique. As an information storage and a means of asynchronous communication, the system would help to collect definitions ("What is the Enterprise Model?"), give better shape to fuzzy concepts and phenomena ("Who are the stakeholders in an Enterprise Modeling project?"), distinguish hot spots ("What are the advantages and disadvantages of modeling in groups?"), gather positive and negative experiences ("What should we do if the modeling participants are not interested in the modeling activities?") of using Enterprise Modeling. Question answering could assist learning of Enterprise Modeling. The intention resulted in the system[23] with about hundred FAQs in its database.

In the above application of FAQ AS, the aspect of organizational memory is emphasized. Still, even if the information itself has no crucial importance for the organization, FAQ AS does automate the question answering procedure. It is convenient to get a question answered from any Internet-connected computer at any time.

Teaching

The initial users of the instance of FAQ AS on HTML[24] were students taking a course on Internet application standards and protocols. The intention was to test both the system's performance and the users' response to such a system.

The assumption regarding benefits from FAQ AS in teaching is as follows. As a course is delivered and assignments given, different students certainly have similar questions which can be answered by the system having an appropriate FAQ collection. Not only the availability of the system is an advantage. Shy students may feel more comfortable talking to a computer rather than a teacher. Reasonable privacy of the communication can lead to a more honest conversation: no one is afraid to look incompetent in the eyes of fellows and teachers. On the other end, teachers can read the logged questions and find out what is poorly understood or seems more interesting. The FAQ database is easily customizable, and the issues of major interest can be quickly added. A user can be addressed personally if he or she has left the e-mail address. The logged user's IP address and time of transactions designate a chain of questions from the same user without revealing the user's identity. Knowing the typical chains may help the teacher prepare the course better.

---

[23] http://ekd.dsv.su.se/faqs.htm, valid in November 2001

[24] http://www.dsv.su.se/html/, valid in November 2001

Obviously creating and maintenance of such a system is rational only if it can be used for the same or similar courses repeatedly.

The impact of FAQ AS on the teaching process was not studied because it required more significant involvement of the teachers and students. Nonetheless, a database with about hundred FAQs was created. The system received about 550 user questions. Although some of the questions seemed genuine, no doubts many of them were asked because of sheer curiosity in order to try out the system.

Customer Service

Preliminary contacts with web-design firms have shown certain interest in adapting FAQ AS for automated question answering in Internet-based customer services. There are no such implementations of FAQ AS yet.

Let us consider a tax authority. Many countries have complicated regulations for local and federal taxation for individuals, their property and small businesses. The regulations are arranged into large documents used mostly by professionals in tax legislation. People call the experts to clarify details. Telephone service tends to have business hours, queues, as well as confusing and slow menu-based redirections. A busy telephone service requires large staff. It is possible to tackle the problem differently. If we split those large documents into question-answer pairs, then a system like FAQ AS can answer customer questions online with no business hours, no queues, and possibly just one employee who maintains the system.

Troubleshooting

Troubleshooting is a special kind of customer service. Most software and hardware manufacturers have web-based information support for their products. The troubleshooting pages there typically have “known bugs” and “report your problem” sections. When people install the software and find out that things do not work, they often consult these pages.

An FAQ list is very appropriate for troubleshooting: it identifies “known bugs” and tells how to fix them. On the web, there exist many FAQ lists on various software and programming issues. None of them is complete though. Major software products and a few programming languages have related Usenet newsgroups. People post their questions to the newsgroups, and peers answer them. A number of typical questions are posted over and over again, therefore some newsgroups have their own FAQ lists, such as for comp.lang.c++[25]. Before someone posts a question to this newsgroup, the person is recommended to consult the associated FAQ list; maybe the question is already answered.

[25] http://www.parashift.com/c++-faq-lite/, valid in November 2001

Some companies put reasonable efforts into maintaining FAQ lists on behalf of their products. At the moment of writing this article Borland Delphi had more than 1400 question-answer pairs in the "Delphi Technical Information Docs" and "Delphi Frequently Asked Questions"[26]. The order of the FAQs was chronological on the website and none in the downloadable files. We can imagine what it takes to find information among so many chaotically listed FAQs.

An FAQ answering system could facilitate the communication between software manufacturers and numerous users of their products. A "gap list", like one that Auto-FAQ had, would augment the system with the benefits of Usenet newsgroups. We should note, however, that FAQ AS does not perform quick indexing of large existing FAQ lists. The AI question-answering solutions that use machine learning are more appropriate in such cases. FAQ AS is convenient for original gradually evolving FAQ collections.

Administration and Maintenance of the FAQ Database

So far only two not too busy instances of FAQ AS have been tested. Therefore administration of the FAQ database is discussed keeping in mind a small to medium size system with 100-400 FAQ entries in the database and 1-200 user queries per day. The administration routines could be more complex for a large and busy system which operates, say, 1 500 FAQs and receives 10 000 user queries per day.

Creating and Maintenance of the FAQ Database

As a new instance of FAQ AS was established, the initial FAQ collection was created before the operation of the system began. No one would ask any questions to an empty database. After the initial collection was ready, the system was tested by tolerant users. At this stage many questions were likely to remain unanswered because either the right FAQs were missing or the system failed to retrieve them. After fine-tuning was done, the system was announced to a broader audience.

If the FAQ database is not based on an existing FAQ list, it may be difficult to figure out candidate FAQs before any user questions are actually asked. One suggestion how to find them is to extract main topics from a relevant manual. Usually candidate FAQs explain basic concepts within the knowledge domain (what-is-this type of questions) and main relationships between them.

There is not much to say about the user information in FAQ entries. Each FAQ should focus on one problem. A longer compound answer should have a menu on the top. Examples are helpful. In the end of the answer, a few links to related FAQs and external relevant sources of information are recommended.

---

[26] http://community.borland.com/delphi/, valid in November 2001, follow "TI's" and "FAQs"

It is important to note that the FAQ database will require continuous maintenance as long as the system is active. Daily examination of transaction logs is the required minimum. The logs are the main source of maintenance information. They contain submitted questions and the system's replies, user IP addresses, time of each transaction, optionally user e-mail addresses, etc. If an on-topic user question is poorly answered, the administrator creates or modifies FAQ entries and incorporates the question into the database. It is a courtesy to notify the user about this new information in the database if he or she has left the e-mail address. More often the updates take place while the system is new. A well-established system with a rather complete FAQ collection and well-drilled keyword sets is fairly autonomous.

Because a natural language based interface does not impose any restrictions on user queries, the users may challenge the system by questions outside its present knowledge domain. The instance of FAQ AS on HTML started with the focus on HTML and basic CGI and Java programming. Nonetheless, the users asked also about XML (Extensible Markup Language), ASP (Active Server Pages), Perl, details about the Apache server, etc. There were also numerous irrelevant questions like "What are you doing tonight?"

A natural language based interface can make us discover that some people use the information system in order to search for information different from what was initially considered. Generally such a discovery positively influences the completeness of the information rendered by the system and broadens its knowledge domain. If, however, the new issue is not to be elaborated in this FAQ collection, an appropriate FAQ entry can redirect interested users to a relevant external information source.

Manual Routines vs. Automated Routines, Tool Support

A question-answering system is completely autonomous when it answers user questions. Yet it does require human input in order to populate its knowledge base. Today's technologies cannot communicate the meaning of a natural language sentence from the human mind to the computer without any manual work done at some point of the development of the system itself or its imported components. The question is how to accomplish this task most efficiently.

Traditionally question-answering systems is endeavor of AI. Nonetheless, high qualification and rare skills of the personnel is the main reason why the solutions of AI are not acceptable in many organizations. In FAQ AS, intelligent decision making is moved from the process of FAQ retrieval to the process of FAQ maintenance, which permits human reasoning. Therefore FAQ AS has *add-on automatization* of administration and maintenance of the FAQ database, unlike AI question-answering systems that have built-in automatization of learning and question answering. Add-on automatization complements human reasoning. It is

flexible with respect to the skills and preferences of the owner of the FAQ answering system. People with the background in Information Retrieval may prefer statistical methods of analysis of the FAQ database in order to find context dependencies between synonyms or to discover similar FAQ entries. People with the background in AI may choose machine learning for semi-automatic creating and maintenance of the FAQ entries. Computerized tools – dictionaries, grammar analyzers, spelling checkers, you name it – are welcome as long as human supervision ensures the quality of the information. The result of the efforts – an FAQ entry – is important whatever methods are used to create it. We can imagine FAQ entries as pieces of conserved human intelligence containing pre-made decisions applied upon a user's request when a query is submitted. This is different from the approach of AI where the computer is taught to make decisions upon a user's request.

As to the existing two instances of FAQ AS, their FAQ entries are created manually using the administration tool and the thesaurus of Microsoft Word. When a new FAQ entry is created, the system's administrator (i) analyzes the FAQ, (ii) considers alternative forms of the question, (iii) selects required, optional, and forbidden keywords for each alternative form, i.e., each auxiliary entry, (iv) selects context dependent synonyms and their appropriate grammatical forms for each keyword. This is how the system learns natural language sentences and their meanings.

Still, there are automatic maintenance routines that do not require reasoning. FAQ AS has an administration tool with such no-frills facilities as "search for an FAQ entry" with various options, "find-replace text", buttons and hotkeys for inserting HTML tags into text. Management of the synonyms for the keywords in FAQ entries is a more elaborate semi-automatic procedure:

- A thesaurus suggests numerous synonyms to any word. Usually no more than $^1/_3$ of ordinary language synonyms are appropriate in the context of a particular FAQ. The administrator selects the right ones.
- Synonyms vary in different contexts. For instance, in "What is the role of HTML on the Internet?" and "Where and why do we apply HTML?" the words "role" and "apply" are synonyms. It is easier to handle such context dependencies if we see previous occurrences of synonymy. By pressing a key, the administrator can see synonyms of a particular word in the context of every FAQ where this word is relevant.
- Repeating word groups are stored in a special repository and reused in FAQ entries wherever necessary without rewriting them.

The aim of automatic routines is to save monotonous work and to help the administrator obtain and filter out information so that he or she can make final decisions.

Reuse of similar entries is an important means of creating new entries "automatically": we copy an old entry, change a few keywords there, and a new entry is ready.

Administrator

The main duties of the administrator of FAQ AS are to monitor transaction logs, find out information gaps in the FAQ collection, discover malfunctioning FAQ entries (i.e., the system failed to retrieve an existing relevant entry), and consequently create new entries or update existing ones.

What qualities should the system's administrator possess? First of all, the administrator must be able to create FAQ entries. The person must have the feeling of the language (i.e., English) and know the subject of the FAQ collection. The administrator must understand how well the FAQ collection covers the subject and where the information gaps are. Generally, the natural language specific part of the work is more advanced than an average computer expert is used to, while the computer specific part should pose little problems to a linguist familiar with computers.

The instance of FAQ AS on HTML was set up by a master student with assistance of the developer of FAQ AS. The master student had never worked with question-answering systems before. It took about five weeks for him to get acquainted with the FAQ answering technique and to populate the initial collection with about fifty candidate FAQs. After the initial testing of the FAQ collection, it took five more weeks to acquire additional fifty FAQs and improve existing ones. It should be noted that those 10 weeks for 100 FAQ entries included learning of the technique, searching for candidate FAQs, and composing of all the FAQ answers. Search for the information for candidate FAQs and FAQ answers proved the most time consuming and tedious task.

An expert can create and test an FAQ entry in 5-15 minutes given the FAQ and its answer (i.e., no search for the information and composing of the FAQ answers is done).

Providing Service vs. Providing Software

FAQ AS and its question-answering technology were designed keeping in mind ordinary websites such as those belonging to university laboratories, software developers, etc. One person can set up a new FAQ answering system in a short time and gradually augment its FAQ collection after the system is put into operation.

The two existing instances of FAQ AS are running at one university department, and the same person is in charge of both of them. This brought up another idea of how to operate FAQ AS: instead of teaching many people how to administrate many FAQ answering systems we can concentrate the maintenance efforts of many analogous systems in hands of a few skillful administrators. Administration of one

instance of FAQ AS does not require much time unless the knowledge domain rapidly grows, yet it does require certain skills. By this approach, the software of FAQ AS is not a desktop application to be distributed to the end users. Rather, the software is to be used by professionals who provide FAQ answering service to the end users.

Performance of FAQ AS

Apparently the most important quality of a question-answering system is how well it answers arbitrary user questions. 480 on-topic questions asked to the instance of FAQ AS on HTML were inspected. The system managed to answer 294 of them, which is 61%. Yet we have to make clear the meaning of those 61%.

Two crucial parameters influence the performance of FAQ AS – completeness of the FAQ collection (a user question can be answered only if the corresponding FAQ exists in the database) and quality of the FAQ retrieval technique (the system must find the corresponding FAQ which does exist in the database). 39% of user questions were not answered mainly because there were no corresponding FAQs in the database at the moment of asking the question. Still, let us remember that the FAQ collection is evolving. In the very beginning the system is able to answer only few questions, but it becomes smarter over time as more user questions are asked, recorded, analyzed, new FAQ entries in the database created and existing ones modified. Then, if we ask the same relevant questions again, the system can answer them. Completeness of the FAQ collection is ever improving, although the collection will never be finished. The universe of conceivable questions of any reasonably broad topic is infinite.

The system's ability to retrieve relevant existing FAQs is determined by the effectiveness of Prioritized Keyword Matching. In Information Retrieval there are two parameters to measure such effectiveness – recall and precision. *Recall* R characterizes the system's ability to retrieve all the relevant items existing in the collection of documents (i.e., FAQs):

$$R = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of relevant documents in collection}} .$$

Precision P characterizes the system's ability to retrieve only relevant items:

$$P = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of documents retrieved}} .$$

In order to measure recall and precision, only those queries were considered that were not duplicate (e.g., "What is CGI?" was submitted and answered seven times, only one was counted), had no spelling mistakes (the system has no spelling checking yet), and had the corresponding FAQs in the database at the moment of asking the question. Thus 259 queries were selected. The measurements for FAQs

retrieved as close answers (i.e., related FAQs – see Figure 2 – were ignored) showed recall 81% with precision 98%.

We should observe that these figures are not static. If the system fails to match a user question to the corresponding FAQ entry, the entry is modified so that next time it does match this and similar questions. Well-drilled entries accumulate past experiences, therefore their ability to match appropriate and ignore inappropriate user questions increases.

We should observe also that the quality of FAQ retrieval very much depends on the experience and skills of the system's administrator who creates and maintains the FAQ entries. The instance of FAQ AS on HTML with the above recall and precision figures was maintained chiefly by a student who had no previous experience in question-answering systems.

Keyword-based search may help to deal with unanswered questions. If the system fails to answer a question, it can transform the question into an appropriate keyword query in order to retrieve related information. This feature is not yet implemented.

Further Research

There are a number of possible directions of development of FAQ AS. One such direction is development of the Prioritized Keyword Matching technique. The latter was originally developed for FAQ answering in English. We may need to adapt the FAQ retrieval technique to a language different from English, especially because FAQ AS is developed in Sweden.

Another problem domain is answering of longer queries. When people encounter a complex problem and search for a solution, often they describe the problem with 100-200 words. The current system has been designed and optimized for answering of one-sentence-long questions.

One more direction of the development is transformation of FAQ AS to a natural language interface of an external relational database. The data structure of such a database represents a limited number of concepts. Hence, there are a limited number of question types that can be asked about these concepts and their relationships. Hence, we can create a limited number of question templates (i.e., FAQs) that cover these question types. Each template can have an associated SQL query that creates the answer to the question expressed in the template. Further research should tell us how to analyze the conceptual data structure of the relational database in order to create the question templates and associated SQL queries.

There is a need to improve the system's administration tool as a decision support tool and to further automate the administration routines as it was discussed previously.

Conclusions

FAQ AS is an information system that answers natural language questions from its own FAQ collection. We can perceive the system as an "answer garden" in an organization. The system's FAQ collection accumulates past experiences in the organization as problem statements (i.e., FAQs) and their solutions (i.e., FAQ answers). FAQ AS allows saving experts' time devoted to answering the same questions over and over again. The system is available at any time from any Internet-connected computer. Reasonable confidentiality of the conversation eliminates the risk to appear incompetent in the eyes of human experts. The FAQ collection evolves as more questions are asked, recorded, analyzed, and new FAQs added.

FAQ AS has an easy to use interface: people can ask natural language questions as well as perform keyword-based search through the FAQ collection.

A typical domain for an FAQ answering system has a large number of complex definitions, concepts, and their interactions. The concepts have very few instances. Two examples of a typical domain are technology and governmental institutions. Domains not appropriate for an FAQ answering system have a large number of data instances typically stored in a relational database. For instance, a timetable.

FAQ AS has been tested using two FAQ collections, one on Enterprise Modeling and the other on HTML. At the current stage of the development of the system, the FAQ entries are created by the system's administrator applying "manual" human reasoning. Still, computerized tools assist with automated maintenance routines. Unlike the AI question-answering systems that have built-in automatization of learning and question answering, FAQ AS has add-on automatization of administration and maintenance of the FAQ database. Add-on automatization complements human reasoning. It is flexible with respect to the skills and preferences of the owner of the FAQ answering system.

The power of FAQ answering by using FAQ AS rises from the fact that rather high quality of FAQ retrieval is achieved by a simple means, which can make automated FAQ answering affordable for an average website.

PART 4

AUTOMATED QUESTION ANSWERING BASED ON THE INFORMATION SYSTEM'S ENTITY-RELATIONSHIP MODEL

1 Introduction of Question Templates

The initial framework of this research was automated FAQ answering. The vision was to build a question-answering system that could be maintained by nonexperts in Natural Language Processing (NLP) and that would be easy to adapt to a new knowledge domain. Then, in the second half of 1997, no one guessed that three years later a simple FAQ answering system would develop into a natural language question-answering interface for an SQL database.

Parts 2 and 3 of this thesis describe an FAQ answering technique. By using a Web-browser, the user submits his or her question to the system. The system receives the question and searches through its database in order to find pre-stored FAQs that correspond to the question. The system recognizes different formulations of the question. After one or several, if any, relevant FAQs are found, the system sends them and their answers back to the user.

When typing the question, the user knows nothing about the structure of the database and the matching algorithm. Each FAQ entry in the database contains a set of keywords that distinguish the FAQ. When searching for relevant FAQs in its database, the system matches these keywords to the submitted user question. The quality of FAQ retrieval is high because FAQ entries are created manually. The question-answering technique is simple and depends mostly on the human ability to select keywords and alternative formulations of a question. The system is portable because it is ready to answer questions with the first FAQ entry in its database.

The Prioritized Keyword Matching technique was first implemented in two FAQ answering systems. Their subjects are Enterprise Modeling[27] and HTML[28]. Salut – a European Union 5th Framework research project which maintains a Web-site about eating disorders[29] – has started deployment of equivalent FAQ answering systems in English and Swedish. There are plans for French and Spanish.

Still, the demand for natural language interfaces on WWW goes beyond automated FAQ answering. Organizations are interested in having question-answering interfaces for structured (e.g., relational) databases. During the later stage of testing the HTML FAQ answering system, the developers of the system were contacted by Metamatrix[30], a company that had developed the database on events in Stockholm, the capital of Sweden. The database[31] is operated by Stockholm Information Service (SIS). Metamatrix wondered whether or not it would be possible to adapt

[27] http://ekd.dsv.su.se/faqs.htm, valid in November 2001

[28] http://www.dsv.su.se/html/, valid in November 2001

[29] http://www.salut.nu/, valid in November 2001

[30] http://www.metamatrix.se/, valid in November 2001

[31] http://www.stockholmtown.com/events/9/index.asp, valid in November 2001

the FAQ answering technique for the SIS database – an SQL database having well-structured data. The first answer was “probably no” because FAQ answering techniques deal with static FAQ entries whereas an SQL database requires dynamic queries that represent all the variety of its data instances. Nonetheless, further reasoning lead to the initial ideas concerning dynamic FAQs having variable parameters that represent data instances in an SQL database.

1.1 Dynamic FAQs

The FAQ answering systems, discussed previously in this thesis, are considerably simpler than their NLP counterparts because they do not comprehend user queries. Nonetheless, they are effective only in domains where concepts and their relationships have rather few data instances. No one is likely to create a separate FAQ entry for each instance in a large data collection.

We can, however, benefit from the simplicity of automated FAQ answering in the task of creating a question-answering interface for a relational database. Let us introduce the notion of a *question template* – a dynamic FAQ as opposed to the traditional static FAQ. A question template is a question having entity slots – free space for data instances that represent the main concepts of the question. For example, “When does <performer> perform in <place>?” is a question template where <performer> and <place> are the entity slots. If we fill these slots with data instances that belong to the corresponding entity of the information system, we get an ordinary question, e.g., “When does Pavarotti perform in Globen?”

The question template’s “answer” is created by help of a *database query template* – a formal database query having free entity slots for data instances, primarily the primary keys. After the slots are filled, the template becomes an ordinary executable database query. For example,

```
SELECT start FROM timetable
WHERE timetable.eventid IN
( SELECT eventid FROM event
  WHERE event.placeid = <place-primary-key>
  AND event.eventid IN
  ( SELECT eventid FROM performer
    WHERE performer.performerid = <performer-primary-key>
  )
)
```

could be the query template associated with the above question template.

Processing of a query template and execution of the query returns raw data which needs to be formatted and complemented with wrapping text such as a header and footer. The wrapping text may contain free entity slots. Both the query template

and the wrapping text form the *answer template*.

1.2 Question Templates and Entity-Relationship Diagrams

The entity-relationship (ER) diagrammatic technique [Chen 1976] is a graphic way of displaying real world concepts, their relationships, and attributes. A written natural language sentence is another way of displaying real world concepts, their relationships, and attributes. By a process called conceptualization, information is transformed into sentences, sentences are transformed into elementary sentences, and elementary sentences into object-role pairs. A conceptual schema describes which elementary sentences may enter and reside in the information system [Nijssen 1977]. Because an ER model, which is a conceptual schema, embodies a set of elementary sentences, we can map a natural language question into an ER model. Figure 1.1 shows an example of such a mapping: each concept and relationship expressed in the question has a counterpart in the ER model.

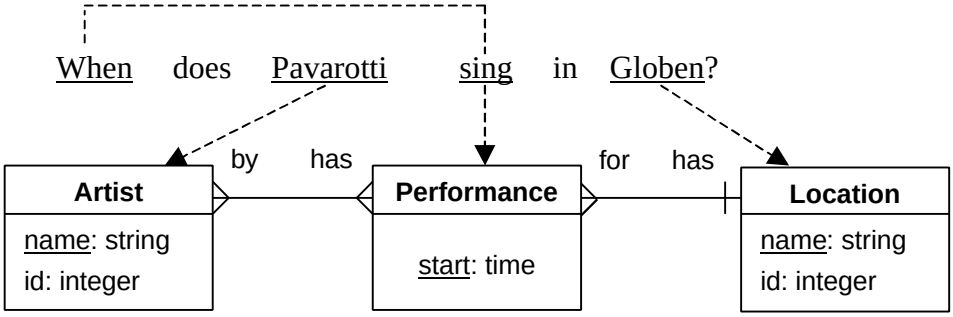

Figure 1.1 Natural language sentence mapped into an ER model.

An FAQ collection covers a certain knowledge domain with a number of questions and their answers. An ER model represents a knowledge domain. Because the ER model of a stable information system is rather static, we can cover the model with a number of question templates – "frequently asked questions" – that represent the entities, their relationships, and attributes in the form of natural language sentences.

[Chen 1983] proposes eleven rules how to construct an ER model reading its free text specification. A reverse operation must be possible too: by reading an ER model, we extract a number of simple questions that represent the concepts, their relationships, and attributes (in fact, [Dalianis 92] proposed a tool that generated pieces of natural language text associated with a conceptual model). Further, we can assign a database query template to each of the extracted questions. Now we have a set of question-answer pairs (i.e., we obtain the answer by executing the query). The system answers a submitted user question by finding corresponding question-answer pairs. On the other hand, if the user question includes concepts that have no counterparts in the ER model, the information in the database is obviously not sufficient to answer the question. Of course, there are many more

details to be considered before a real question-answering interface is set up.

1.3 Research Problem and the Main Results of Part 4 of this Thesis

This is a continuous research and should be considered in the context of the previous work. The **research problem** of this part of the thesis is to adapt the FAQ answering technique, presented in Parts 2 and 3, to a question-answering interface for a relational database (called "target database" further in the thesis). The modifications are introduced according to the following guidelines:

- Static FAQ entries are replaced with dynamic question and answer templates, which makes it possible to customize the "FAQs" and include data instances that reside in the target database and are relevant to the user query.
- Prioritized Keyword Matching is preserved as the means of matching a user question to a question template. The algorithm of selecting the question templates – candidates for Prioritized Keyword Matching – is modified to include the entity slots which did not exist in FAQ entries.

The main **research results** in this part of the thesis are the design, implementation, and evaluation of the *question assistant* [32] – the question-answering interface for the SIS database which answers questions concerning events in Stockholm.

An example of asking a question follows. By using a web-browser, the user submits a one sentence long question, as shown in Figure 1.2.

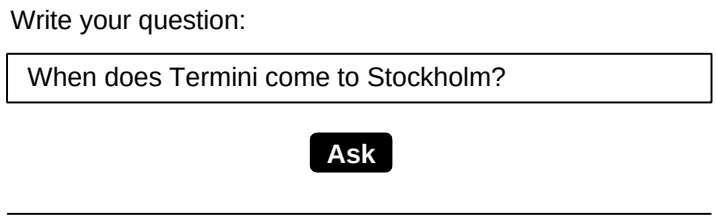

Figure 1.2 Example of a user question.

The question assistant receives the user question, finds out that there are "Termini" and "Stockholm" registered in the database as an performer and a location, and offers one or several, if any, system's own interpretations of the question, as shown in Figure 1.3.

[32] The research prototype is available at http://ekd.dsv.su.se/faq-sis.htm, valid in November 2001

Your question: When does Termini come to Stockholm?

Question assistant understood your question as:

Ask **Has** Claudia Termini **any performances in** Stockholm**?**

Answer to the following question may be interesting too:

Ask **What goes on in** Stockholm **during the period** 4 June 2001 - 2 September 2001**?**

Figure 1.3 First reply to the question – interpretations of the submitted user question.

The user selects an interpretation and clicks on the "Ask" button. The system receives a new query – the selected interpretation – and answers it. The answer is shown in Figure 1.4.

Has Claudia Termini any performances in Stockholm?

Yes, *Claudia Termini* performs in *Stockholm*.

Time	Place
2000-08-09 20:00	Oscar's Church

Figure 1.4 Answer to the question.

The answer of the selected interpretation is supposed to be also an approximate answer of the original user question.

The requirements, design, and algorithms of the old FAQ answering system change as the environment of the system changes. What are the differences between the old and new systems?

The question-answering principles of the new system remain the same whereas the requirements for the user query and the output are different because the new queries and answers may refer to data instances in the target database. Further in the thesis, we define the appearance and scope of valid user queries. Requirements for the target database are new in this research because such a database did not exist in automated FAQ answering.

The new design considers issues such as linking the entity slots in the question templates to the information system's entities and further to the data instances in

the target database. The system has an index of the data instances for fast access.

The new question-answering algorithm uses question templates and creates interpretations of the user question with respect to the data instances referred to in the user question rather than retrieves static FAQ entries from the FAQ database. The execution time of the algorithm depends on the amount of data in the target database. Therefor one of our tasks is to evaluate the execution time of the algorithm and to identify the main parameters that influence the execution time.

[Burg 1997, pp. 81-83] argues that there are three specificity levels in an NLP lexicon: the lexicon incorporates application-specific, domain-specific, and common sense knowledge. Such a lexicon has not been able to solve a long lasting problem of NLP – disambiguation of the meanings of words in the context of a short expression. This thesis has proposed the idea of a multiple lexicon (see the chapter “Idea of Multiple Lexicon” in the paper “Automated FAQ Answering: Continued Experience with Shallow Language Understanding” in Part 3 of this thesis). A multiple lexicon consists of a number of small, analogous, autonomous units operated in a question-specific knowledge domain – a refinement of the application-specific knowledge domain. Each question template incorporates such a unit. This research further explores the possibilities of the multiple lexicon.

1.4 Structure and Contributions of Part 4 of this Thesis

Part 4 has eight sections. Section 2 presents requirements, logical structure, and processes of the question assistant. Section 3 discusses its components: entity definitions describe which data instances may fill which entity slots, the data index helps rapidly locate data instances relevant for the submitted user question, question templates match the user question, answer templates answer it, transaction logs show what questions have been asked and what answers given. Section 4 presents a semi-formal description of the algorithms and an estimation of their execution time. Section 5 discusses maintenance of the question assistant, paying special attention to the manual maintenance procedures. Section 6 evaluates performance of the system – its ability to retrieve relevant information and not to retrieve irrelevant information. Section 7 gives an insight into related techniques and systems: Information Retrieval, Natural Language Processing, graphical user interfaces based on the conceptual model of the underlying information system, and Ask Jeeves. Section 8 suggests several directions of the further research and presents the conclusions of Part 4 of the thesis.

2 Requirements, Structure, and Processes of the Question Assistant

The question assistant receives a user question, responds, and logs the transaction. It does not do anything else. This section presents the requirements and an overview of the components and processes of the question assistant.

2.1 Requirements for the Question Assistant

Besides the overall appearance of the system, we distinguish also the requirements for the user input and output, as well as the target database.

2.1.1 What the Question Assistant Is and Is not

There are a few considerations that determine the appearance and functionality of the question assistant. It is important to be aware of them in order to have the right opinion about what the question assistant is and why it is like this.

The question assistant is not the only interface of the target database, but rather a *supplement to a large and tedious input form*. Unlike a rigorous form, the question assistant is allowed to give imperfect answers. Its average recall (ability to retrieve all the relevant information available in the database) and precision (ability to retrieve only relevant information) are expected below 100%. Still, there is always an option: the user fills in the input form if he or she is not satisfied with the answer rendered by the question assistant.

The question assistant is a *descendant of an FAQ answering system*. It does not answer a user question directly but rather

1. finds a number of appropriate question templates, if any, and creates interpretations which approximate the original user question;
2. answers the user selected interpretation using a database query template (the system does not generate database queries).

Our question-answering approach does not yield better quality of the service than the traditional approaches of NLP based on semantic analysis of user queries and subsequent generation of database queries. The *competitive edge of the question assistant is its simplicity* while rendering still acceptable quality of the service. The question assistant is designed for organizations that do not want to use complex and expensive systems. We have a hypothesis, based on the previous limited experience, that an expert can install a new instance of the question assistant for a medium size relational database within a few weeks. For comparison, it takes between 6 and 12 person-years to develop an Internet search engine [Tångring 2001, p. 32].

The very first (rejected) paper that described the question assistant got a thoughtful review:

> "The main problem with the system you describe is, in my opinion, that people who access a DB will, in all likelihood, want to get precise answers (since DBs are used to store precise information), and under these circumstances your approach, which may give approximate answers, is likely to be unsatisfactory for the user. The additional expense of implementing a system that properly translates Natural Language queries into SQL (or similar) cannot be avoided in such a case. Such systems have been developed in the past, as you will probably know, but their success as exclusive means of accessing DBs has been quite limited (it seems that users prefer multimodal access systems: QBE or graphical tools to get a first impression of what the DB contains, SQL for a first round of queries, Natural Language for modifications of the original queries assuming that the user generated SQL queries are translated into Natural Languages by the system."

The reviewer is right that most users would choose a question-answering interface that gives a precise answer rather than an approximation similar to an FAQ. Most users would eagerly choose an interface that combines natural language queries with graphical tools of information representation. They would choose if there were any broad selection of choices. Unfortunately, the research in NLP done over several decades has not managed to overflow the Internet with natural language interfaces. Public question-answering systems are still rare today. According to our experience, many users are pleased with any reasonably good opportunity to make the computer "understand" plain English.

The question assistant has its background in the information systems' research, therefore its solutions are understandable for information systems' people. These solutions are based on conceptual modeling [Loucopoulos and Zicari 1992] of information systems and use a property common for many natural language sentences and all entity-relationship models: they embody the structure "object-relationship-object-relationship…" The goal of this research is *not* to confirm or deny any popular language processing theory but rather to try another approach to building question-answering interfaces.

Finally, the question assistant is *not* a Web indexing tool although its user interface is Web-based. The system answers questions using data in a relational database. Therefore text retrieval from a large collection of free text documents, as in Information Retrieval, is *not* the research problem of this thesis.

2.1.2 Requirements for the User Input

The only user input is a one sentence long question in plain English. Although the user is allowed to type whatever he or she wants, the question assistant assumes

that user questions comply with certain rules.

Explicit Questions

The information in the user question must be explicit and self-contained. The question assistant does not keep track of the dialog and is not aware of the previous questions. The system does not deduce new facts from those given in the question. The only cases where assumptions are made are (i) default time references and (ii) when the user submits keywords (see "Time References" and "Keywords" further).

References to Data Instances

A user question may contain references to data instances, such as textual names of real world objects, residing in the target database. The name of an object should not exceed one sentence. For example, the question in Figure 1.2 contains the words "Termini" and "Stockholm". The question assistant recognized them as the singer Claudia Termini and the Stockholm city.

References to Generic Concepts

The user question may refer to generic concepts, such as "castle", "jazz", "going out", etc., not only to specific names of events or people. Figure 2.1 show two examples.

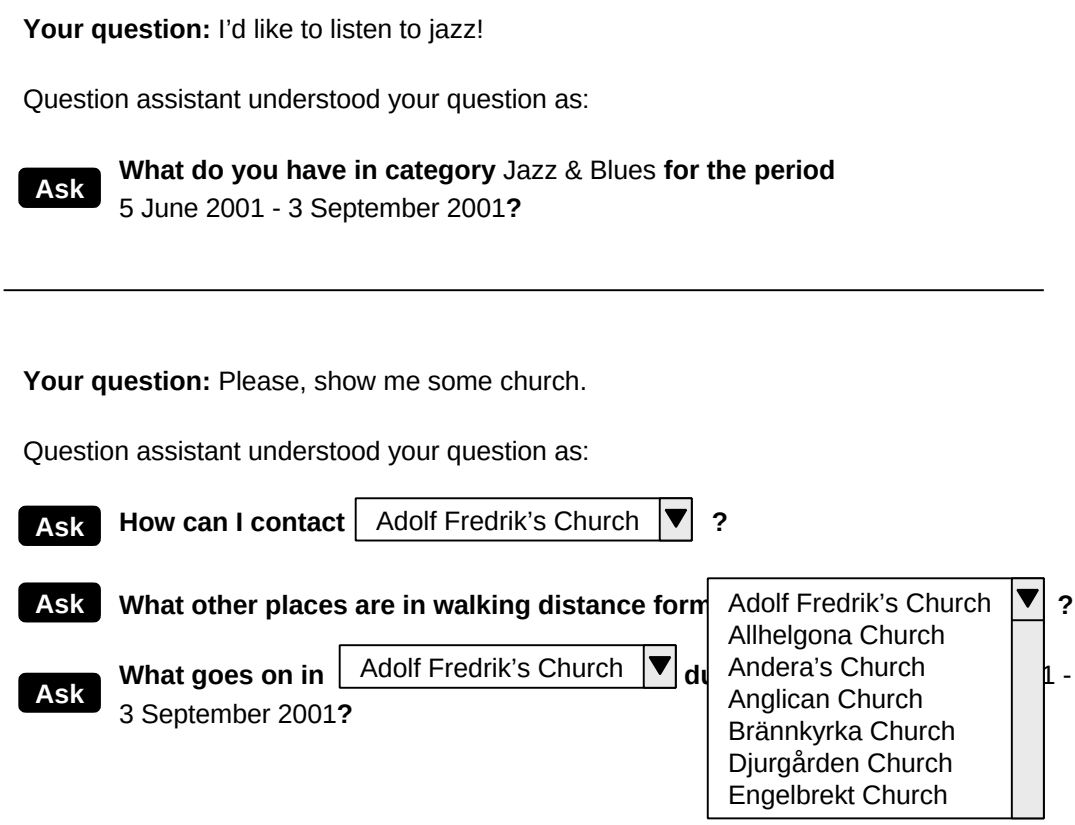

Figure 2.1 Replies to the inquiries about jazz and churches.

Time References

The question assistant answers questions about events, and events happen in time. The question assistant recognizes two kinds of time references in the user question: (i) a day and (ii) a period between two days, including the first and the last indicated day.

A day is indicated as follows:

- A date having several formats, such as “5 June 2001”, “June 5, 2001”, “5 Jun 2001”, “2001-06-05”, “05/06/01”. Only the name of the month may be written by letters.

  If the date is written as slash-separated numbers, the following formats are accepted:

  - if the years has 4 digits, the formats are “`dd/mm/yyyy`” and “`yyyy/mm/dd`”;
  - otherwise the formats are “`dd/mm/yy`”, “`mm/dd/yy`”, “`yy/mm/dd`”, applied in the given order.

  If the date is written as hyphen-separated numbers, only the formats “`yyyy-mm-dd`” and “`yy-mm-dd`” are accepted.

- Common words such as “tonight”, “tomorrow”, etc.
- Weekday references such as “Friday” with possible modifiers “next”, “last”, or “previous”.

A period is indicated as follows:

- Any two day references separated a hyphen, “to”, and “until”, or wrapped into “between `<day>` and `<day>`”, “between `<day>` – `<day>`”, etc.
- Common phrases such as “this month”, “next week”, etc.

The above selection of time formats is not exclusive. Studies of user-preferred formats should be done during the operation of the question assistant, and new formats added as needed.

Default time period. Many user questions have a temporal aspect without explicitly indicating it. For example, ”What’s on at the opera?” implies the nearest future rather than performances during the entire history of the Royal Opera in Stockholm. If a user question implies time, the default period 90 days from today is assumed. There is no default day, only a period. Figure 2.2 shows an example with the default time period.

Numbers

The question assistant recognizes numbers in a user question. Still, there has not been much use of them so far. For the sake of simplicity numbers are ignored in

further processing of a user question.

Keywords

Our experience shows that users tend to skip typing as much as possible. Therefore, instead of writing a complete question, they often ask keywords. If the keywords represent a *concept*, the question assistant retrieves the question templates that represent general knowledge about the concept. If the keywords represent a *data instance* residing in the database, the question assistant finds out which concept the instance belongs to, and then retrieves the question templates that represent general knowledge about the concept with respect to the given data instance. Figure 2.2 shows an example of asking a keyword.

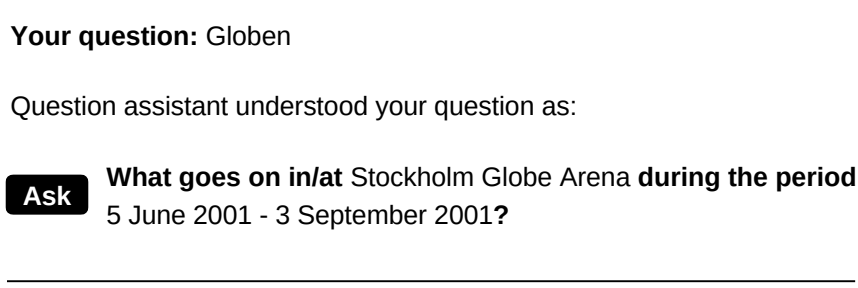

Figure 2.2 Reply to a keyword.

Meaningless Questions

The question assistant uses Prioritized Keyword Matching to match a user question to a question template. This technique cannot distinguish between a meaningless and a meaningful question. In most cases it is not necessary because genuine inquiries are typically meaningful. If the user question is meaningless, the question assistant does its best to find interpretations that resemble the user question and are meaningful.

Misspelled and Unrecognized Words

The question assistant warns the user about the words it does not recognize. The system has a collection of about 9500 truncated words, word stems in most cases, as a stock of correct words it does recognize (see Section 3.2.2).

In the example in Figure 2.3, the Swedish name of the Concert Hall – Konserthuset – is misspelled as “Koncerthuset”. The question assistant replaces it with “konserthuset” and “konsertområde” as synonyms, and proceeds with the new words in the input query.

Your question: Koncerthuset

Spelling. Question assistant adjusted your question to the content of the Events in Stockholm database and replaced '*koncerthuset*' with 'konserthuset' and 'konsertområde'.

Question assistant understood your question as:

Ask **What goes on in/at** Concert Hall **during the period** 5 June 2001 - 3 September 2001**?**

Ask **How can I contact** Concert Hall**?**

Figure 2.3 User question with a misspelling.

2.1.3 Requirements for the User Output

The first stage of answering a user question is creating a number of interpretations, if any, of the question, as shown in Figures 2.1-2.3. Usually the interpretations incorporate data instances from the target database. If the user question contains an *ambiguous reference to a data instance*, the question assistant does its best to resolve the ambiguity and selects a number of data instances that correspond to the ambiguous reference, as shown in Figure 2.4.

Your question: What does Erik do?

It seems the question assistant has no accurate answer.
Would the answer to the following question help you?

Ask **When and where has** Erik Holmström / Erik Saedén / Erik Steen / Erik Westberg / Erik Williams, violincell / Erik-André Hvidsten / Lars-Erik Berenett **performances?**

Figure 2.4 Reply to a question with an ambiguous reference to a data instance.

The second stage of answering a user question is answering the interpretation selected by the user. The user selects the right data instance in the drop-down list, if there has been any ambiguity. Figure 1.4 in Section 1 shows an example of an answer. Each answer consists of two parts:

1. The text of the selected interpretation with the selected data instances, if there have been any ambiguities.
2. The answer of the question formulated in (1). The answer contains the necessary data instances as well as the text that explains them.

2.1.4 Requirements for the Target Database

Not every database qualifies for having the question assistant. Here are the requirements for the target database.

Relational Database

The question assistant was developed as a question-answering interface for a relational database. We have a hypothesis that it should not be difficult to adapt the system for object-oriented databases [Bertino and Martino 1993] because the technique is based on the properties of entity-relationship models of information systems rather than the properties of relational databases.

Appropriate User Audience

An easy to use question-answering interface is supposed to attract people and, thus, distinguish the information system among its competitors. This poses two requirements. First, the information system must have an audience of active *users interested in the subject* it covers. People will not use a question-answering interface if they do not care about the information it renders. Second, the information system must be *easily accessible* by means of widespread communication technologies such as WWW. The question assistant is not efficient if operated as a desktop application used by one person because of the required maintenance efforts.

Textual Names

People refer to object by their names, therefore the target database is expected to have columns that contain text of names of people, addresses, titles of events, etc. that are, in most cases, short pieces of text. For example, in “When does `<performer>` perform in `<place>`?” the entity slots `<performer>` and `<place>` are to be replaced by the text of the names of a person and a place.

Stable Primary Keys

In the world of relational databases, primary key is the unique identifier of an object. Different primary keys identify different objects even if they have the same name. The question assistant counts on that. On the other hand, reassignment of a primary key to another object may lead to obsolete information on primary keys in the internal database of the question assistant and, consequently, to cause the system malfunction. The target database is expected to have stable primary keys.

Consistent Structure of the Text

A column in a table of a relational database contains an array of analogous data instances. Ideally, the data instances have similar structure of their text: groups of words having the same meaning appear in the same order, the same delimiters are used. Table 2.1 shows two sample columns. In one column data instances preserve consistent structure of the text, in the other column the structure is deliberately messed up.

Table 2.1 Examples of consistent and messy structure of the text

Consistent structure of the text	Messy structure of the text
John Smith, the father	Father Smith, John
Mary Jones-Smith, the mother	Jones-Smith Mary, the mother
Lilly Smith, the baby	Lilly “Baby” Smith

Most people would agree that data instances (e.g., person names and addresses) having consistent structure are easier to process. An example of data instances where consistent structure is not possible is titles of events, Table 2.2.

Table 2.2 Titles of events do not have consistent structure of the text

Non-consistent structure of the text
Stockholm Jazz Festival
International Symposium on Myelodysplastic Syndromes
La Mayonnaise Open

Right Formulation in the Right Category

In the SIS database, the names of rock and pop artists are classified as events rather than person names. For example, the concert of Sting is titled “Sting”. If a user makes an inquiry about Sting, the question assistant offers an interpretation “When and where does ‘Sting’ start?” instead of more reasonable “When and where has Sting performances?”

In the target database, objects are expected to be placed into right categories and to have formulations suitable for the chosen category. This improves the quality of the retrieval of data instances and is essential for creating reasonable interpretations.

Right Formulation for the Right Attribute

In the SIS database, there is an event titled “Santa Lucia Concert at the Globe Arena”. The event has the location “Globe Arena” as an attribute, therefore it is not necessary to include the name of the location into the title of the event. Because the latter happens to be the case, the question assistant extracts “at the Globe Arena” as a reference to Globen. Now, if a user submits one word “Globen” as a query, the

question assistant offers two interpretations of the query: the expected “What goes on at Stockholm Globe Arena during the period…?” and an unexpected “When and where does ‘Santa Lucia Concert at the Globe Arena’ start?”

From the question assistant’s point of view, the desired formulation of the above data instance would be:

- title: “Santa Lucia Concert”;
- location: “Stockholm Globe Arena”.

In the target database, descriptions of objects are expected to be split and assigned to the right attributes.

English

The question assistant was created for question answering from a database where textual data instances are in English. In order adapt the system to other languages, further development is needed.

2.1.5 Other Requirements

The structure of the *transaction logs* is discussed in Section 3.5.

The expected skills of the personnel, the *administrator* of the question assistant are discussed in Section 5.

2.2 Structure of the Question Assistant

The question assistant is a mediator between a user and a database. It is designed as a “plug-in” to a relational database, i.e., the same software can be installed and operated for different databases. Figure 2.5 shows the logical design of the system.

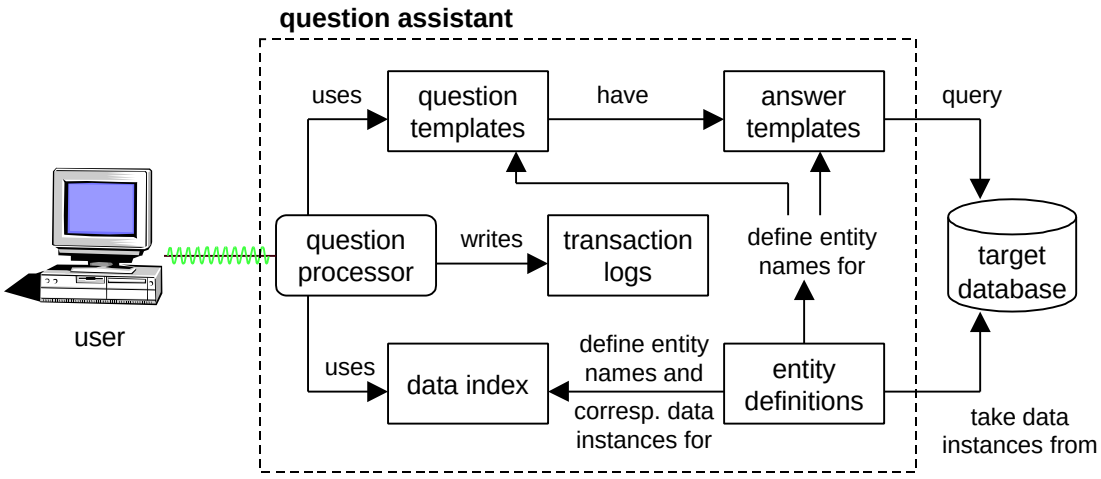

Figure 2.5 Logical components of the question assistant.

- *Entity definitions* describe which entities the question assistant knows and which data instances belong to each of these entities.
- The *data index* is the index of the data instances covered by entity definitions. The index helps to rapidly locate the data instances and their corresponding entities relevant to the user question.
- The system matches the user question to a number of pre-stored *question templates* and retrieves relevant ones.
- Each question template has an associated *answer template* that generates the answer to the user question. The main component of an answer template is its database query template.
- The *question processor* is the main actor attached to an HTTP server.
- User questions and the system's replies are logged in the *transaction logs*.

Section 3 presents more detailed discussion on the components of the question assistant.

2.3 Processes of the Question Assistant

The processes and data flow of the question assistant are shown in Figure 2.6.

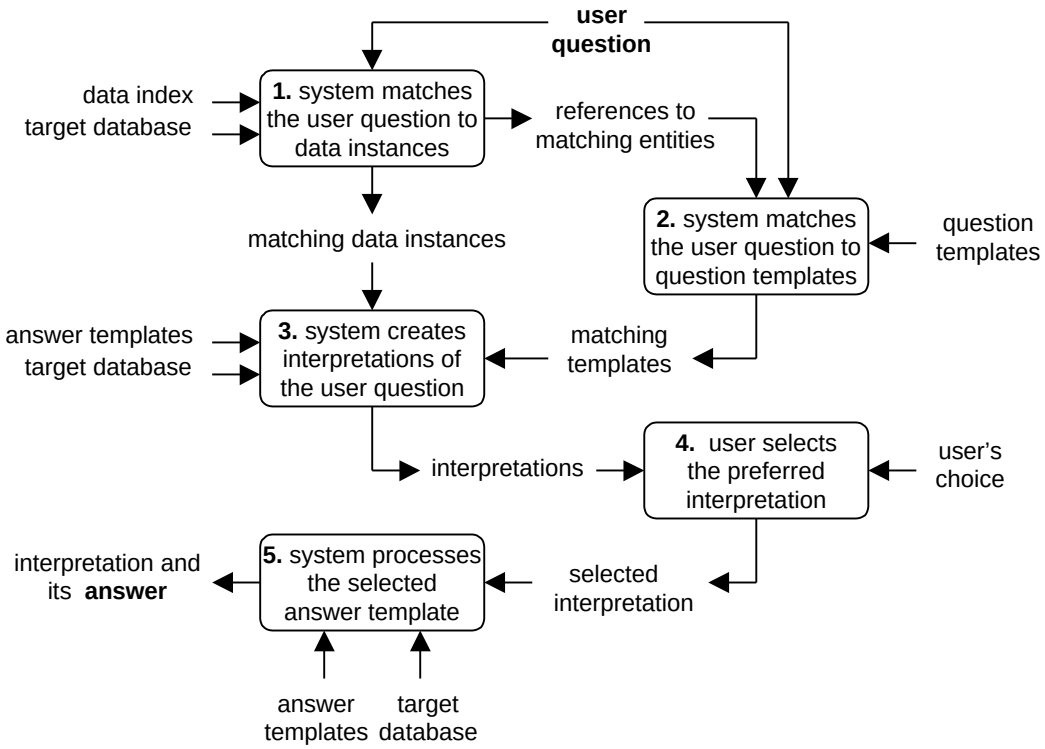

Figure 2.6 Main processes and the data flow in the question assistant.

Process 1. When a user submits a question (example in Figure 1.2), the system identifies the data instances and their corresponding entities represented in the question.

Process 2. The system matches the submitted user question to the question templates and retrieves one or several, if any, question templates that are (i) semantically close to the user question and (ii) reference the same entities as the user question.

Process 3. For each question template received from Process 2, the system fills its entity slots with the data instances from Process 1 and thus obtains a number of interpretations of the submitted user question. The system verifies whether or not the interpretations have any answers. The interpretations that have answers are passed to Process 4.

Process 4. The user selects an interpretation (example in Figure 1.3) and resubmits it as a refinement of the initial question.

Process 5. The system processes the answer template associated with the selected interpretation and creates the answer of the interpretation (example in Figure 1.4). This is also the answer to the initial user question if the system has managed to find the right interpretation of the question.

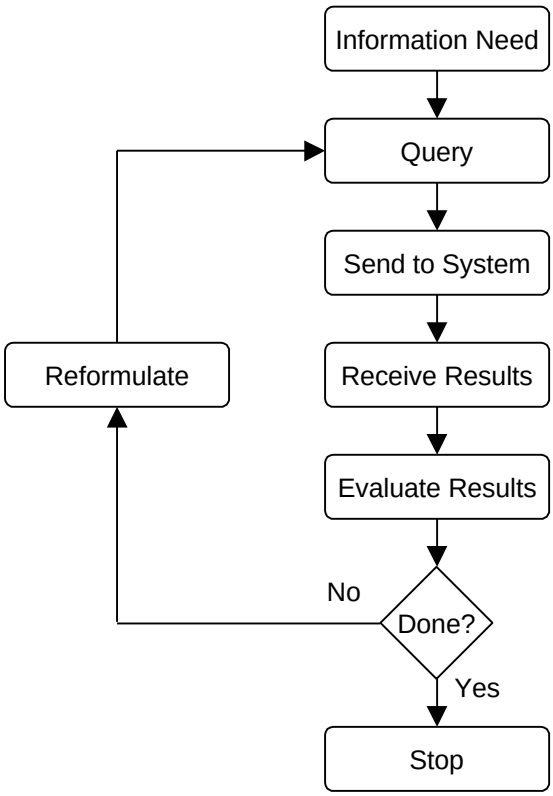

Figure 2.7 Standard model of information access used by Web search engines.

[Hearst 1999, p. 263] presents a diagram of the standard model of the information access process used by Web search engines (Figure 2.7).

The question assistant – a question-answering interface for a relational database – employs the same model. While Figure 2.6 focuses on the processes from the system's point of view, Figure 2.7 focuses on the processes from the user's point of view. Table 2.3 shows the correspondence between both views.

Table 2.3 Correspondence between Figures 2.6 and 2.7

Figure 2.6	Figure 2.7
Input to Processes 1	"Information Need" → "Query" → "Send to System"
Processes 1, 2, and 3	Between "Send to System" → "Receive Results"
Process 4	"Receive Results" → "Evaluate Results" → "Reformulate" → "Query" → "Send to System"
Process 5	Between "Send to System" → "Receive Results"
Output from Process 5	"Receive Results" → "Evaluate Results" → "Stop"

Processes 1, 2, and 3 are responsible for finding the right interpretations of user questions and, therefore, hold the main focus of this research. Section 4 presents a semi-formal description of the algorithms of these processes.

2.4 Summary

Three major features determine the appearance and functionality of the question assistant. First, the question assistant is a supplement to a large and tedious input form rather than the only means of data access for the target database, therefore it is allowed to give imperfect answers. Second, the question assistant has two stages of answering a user question: it (i) creates a number of interpretations, if any, of the user question and (ii) answers the user selected interpretation. Third, the question assistant must be easy to deploy and maintain so that it can compete with NLP systems which create a better illusion of computer intelligence.

A user question is asked in plain English. The question must be explicit, as the system does not keep track of the previous questions and does not deduce any new knowledge from that given in the question. The question may refer to generic concepts, data instances residing in the target database, as well as time. The system corrects misspelled words. If the user question is meaningless, the system finds interpretations that resemble the original question and are meaningful.

The target database is a relational database. The most interesting data instances, from the question assistant's point of view, are textual names of real world objects. These names may appear in user questions when the questions refer to data instances.

The question assistant uses the same model of information access that most Web search engines use.

3 Components of the Question Assistant

This section presents all but one components of the question assistant shown in Figure 2.5 – entity definitions, the data index, question templates, answer templates, transaction logs. The algorithms of the question processor are discussed in the next section.

3.1 Entity Definitions

Entities, also called entity types, are classes of real world objects represented as data instances in the target database.

Entity slots in question and answer templates are the counterparts of the entities of the information system. The user never sees any entity slot explicitly: when the interpretations of the user question are created, the entity slots are filled with data instances (more precisely, names of the real wold objects) as we can see in the examples in Figures 2.1-2.4.

How does the question assistant know which data instance pertains to which entity slot? The links between entity slots and their data instances are established in entity definitions. Each entity slot has a name, the same name as the corresponding entity of the information system. In entity definitions, each entity name is linked to its corresponding data instances by the means of a formal database query. Knowing the name of an entity, the system uses the database query and retrieves the data instances that belong to the entity.

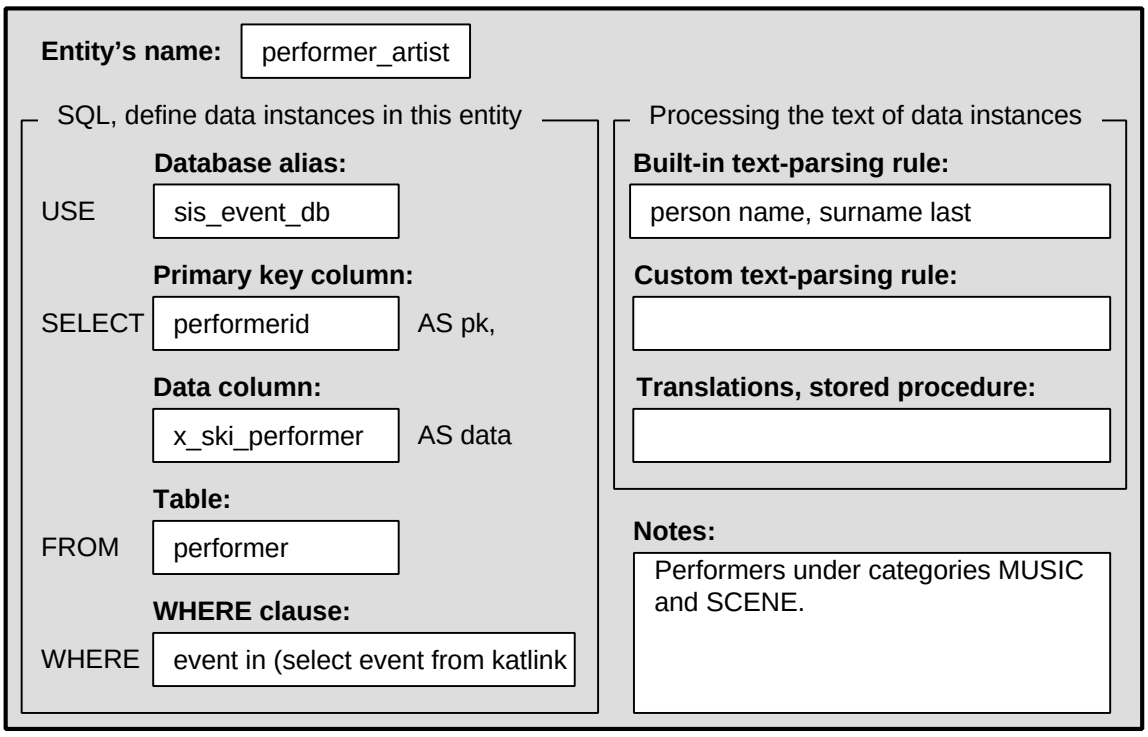

Figure 3.1 Sample entity definition in the edit form.

3.1.1 Constituents of an Entity Definition

Figure 3.1 shows the input fields of the edit form for entity definitions in the administration tool. The constituents of an entity definition are the entity name, SQL query, parsing rule, as well as the name of the translation procedure.

Entity name is the unique identifier of an entity. The entity slots in question and answer templates that reference this entity must carry this name.

SQL query unambiguously defines which data instances belong to the entity. The query is assembled from the database name, table name, the name of the primary key column, the name of the column which contains the names of the real world objects, and the where-clause which identifies a subset of the rows in the table. Figure 3.1 shows the query

```
SELECT performerid AS pk, x_ski_performer as data
FROM performer
WHERE event IN (
   SELECT event FROM katlink WHERE katid IN (
      SELECT 52
      UNION
      SELECT 53
      UNION
      SELECT id FROM kategori WHERE parent IN (52,53)
   )
)
```

Parsing rule in the scope of the entire entity tells how to create a number of synonyms of the text of each data instance that belongs to the given entity. The parsing rule is optional, though very welcome. Parsing rules are discussed in Section 3.1.3.

Translation procedure is optional. The need for translations appeared when both English and Swedish place names were permitted. Let us consider the user question “How can I get to Arlanda?” and the relevant data instances (i) “Arlanda airport” and (ii) “Arlanda flygplats”. Both (i) and (ii) mean the same. In order to avoid duplicates, an entity-specific procedure on the SQL server translates all data instances into English.

3.1.2 Two Components of a Data Instance

The SQL query always retrieves two columns called “pk” and “data”. The column “pk” contains unique *primary keys* (in the example: `performerid AS pk`). The column “data” contains *textual data* (in the example: `x_ski_performer as data`) which needs not to be unique. Hence, every data instance always has two components: the primary key and the text of the data instance. By default, the text of the data instance is meant when just “data instance” is written.

The textual data column contains names of real world objects, such as “The Museum of Modern Arts”, “John Smith”, “Depeche Mode”, “Tyresö”, etc. These names are short pieces of text, usually up to ten words long. The column may contain numbers which are treated as pieces of text. Their numeric values are disregarded.

In an entity definition, unions of tables are not permitted in order to preserve uniqueness of the primary keys. On the other hand, entity definitions may overlap, and one data instance may belong to more than one entity. Overlapping implies that the same data instance may have different interpretations in the context of different entities.

A data instance is globally identified by a unique combination of the entity name and the primary key.

3.1.3 Parsing Rules

After a user submits a question, the question assistant matches a number of data instances from the target database (more precisely, the text of the data instances) to the user question. The user may choose synonyms to represent a data instance in his or her question. In order to deal with such synonymy, the question assistant uses text-parsing rules *applied to the text of the data instances* rather than the user question. A text-parsing rule defines a set of synonyms of a data instance created by processing the text of this data instance. There are four types of text-parsing rules:

- built-in entity-specific text-parsing rules;
- custom entity-specific text-parsing rules;
- x-rules, which are data-instance-specific text-parsing rules; and
- no rules at all.

Built-in Entity-Specific Text-Parsing Rules

It may happen that a large number of data instances that belong to one entity have the same structure of the text, e.g., a list of person names or addresses. Knowing the meaning of each part of the text (e.g., first name, middle name, last name), the question assistant reformulates the entire text and thus creates a number of synonyms of each data instance. The system then matches these synonyms to the user question. It is the entity-specific text-parsing rule that defines the meaning of each part of the text of the data instances that belong to the given entity.

Physically, built-in entity-specific text-parsing rules are implemented in the program code of the question assistant. At the moment of writing this thesis, the question assistant had three built-in entity-specific text-parsing rules that defined three formats of person names: (i) names where the surname is last, (ii) names where the surname is first, and (iii) names where the surname is first and comma

separated. These rules let the system recognize that “John Smith” and “Smith John” are the same person name, that the first name “John” is more ambiguous representative of “John Smith” than the surname “Smith”, they process prepositions such as “von”, “van”, “der”, “de”, “fon”, “di”, “mc”, “du”, etc. Figure 3.1 shows an example where (i) has been selected for the entity “performer_artist”. It is a duty of the administrator of the question assistant to select the right rule so that the system knows which subroutines to apply in order to process the text of the data instances.

Custom Entity-Specific Text-Parsing Rules

Because built-in text-parsing rules are embedded in the program code of the question assistant, it is not possible to change or add built-in rules without recompiling the system. A custom entity-specific text-parsing rule, on the contrary, is denoted by a character string using specific syntax. The question assistant interprets the character string and, for each data instance in the entity, creates a set of synonyms by changing the order of the words and the distance between them. Figure 3.2 shows an example of a custom rule in the entity edit form of the administration tool.

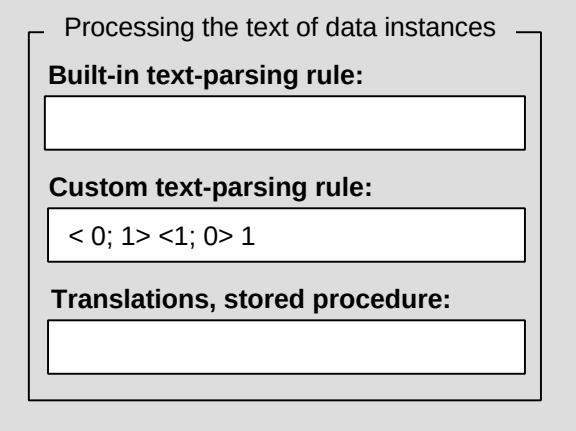

Figure 3.2 Example of a custom entity-specific text-parsing rule.

It is a duty of the administrator of the question assistant to compose a correct custom rule. The syntax of custom rules was chosen simple. The words are referenced by their sequence numbers in the original text of the data instance: 0, 1, 2... The accepted order and distance between the words are indicated using the syntax for processing phrases described in the chapter “Phrases” in the paper “Automated FAQ Answering: Continued Experience with Shallow Language Understanding” in Part 3 of this thesis. If there were no built-in rules, a typical custom rule for person names then would have been “<0; 1> <1; 0> 1”. Having such a rule, “John Smith” gets two synonyms – “Smith John” (introduced by “<1; 0>”) and “Smith” (introduced by the last “1”).

Custom text-parsing rules define purely graphical manipulations with words, therefore they are less powerful than built-in text-parsing rules. They are a “quick and dirty” solution introduced in the first version of the question assistant and are not used anymore after appropriate built-in rules have been implemented.

X-Rules

Unfortunately, general rules tend to have exceptions. Entity-specific text-parsing rules cannot cope with particular synonyms such as popular pseudonyms. Therefore we introduce x-rules (extra rules) attached to individual data instances wherever necessary. X-rules have the same syntax as required, optional, and forbidden keywords in the question templates. The administrator of the question assistant composes x-rules and manually denotes the desired appearance of the synonyms of the data instances. Figure 3.3 shows an example of an x-rule which comprises the name and pseudonym of the famous rock musician.

performerid:	x_ski_performer:	x-rule:
20525	Sting	<gordon*; sumner*> <sumner*; gordon*> sting stings

Figure 3.3 Example of an x-rule.

Whenever the question assistant matches a data instance to a user question, it matches the x-rule if such is defined for the data instance. A data instance may have no more than one x-rule. The total number of x-rules operated by the question assistant is limited only by the number of data instances in the target database. Unlike entity-specific rules, x-rules do not require that data instances in an entity have consistent structure of the text. A disadvantage of x-rules is that, being data-instance-specific, they take a lot of work to be composed for a large number of data instances. If the text of the data instances is not stable, x-rules are not stable either.

No Rules

If the structure of the text of data instances in an entity is not consistent, it is not possible to apply entity-specific text-parsing rules. If the number of data instances is large, it is discouraging to create x-rules for all of them. In the SIS database, this is the case for the titles of events and the names of locations. The question assistant uses a special algorithm that created synonym sets of “ruleless” data instances.

Unlike built-in entity-specific text-parsing rules, the “ruleless” algorithm has little knowledge about the meanings of the words in data instances. First, the algorithm splits the text of a data instance into pieces separated by commas, semicolons, dashes, parentheses, the words “at”, “by”, “near”, “on”, “in”, etc. Then the algorithm drops articles, if appropriate, and combines the words in each piece of so that some combination hopefully matches the corresponding name or title of a real world object. Measurements of the performance of the question assistant (see

Section 6) demonstrate that the "ruleless" algorithm shows mediocre results. Entity-specific text-parsing rules, which incorporate as much entity-specific knowledge as possible, are preferred.

3.1.4 Reusability of Entity-Specific Text-Parsing Rules

Custom entity-specific text-parsing rules are easy to create and modify because they are composed as combinations of sequence numbers of words. Reusability of built-in entity-specific text-parsing rules, which are implemented in the program code, is a more serious issue.

The question assistant has built-in text-parsing rules for processing of person names. These rules were designed having the SIS database in mind. It is not clear yet how easy or difficult it would be to reuse these rules for processing person names in another database. Would there be a need for extensive programming again, or the same routines can be reapplied without modification?

Supposedly, the format of the names of the same sort of things – names of people, places, and companies, titles of events and movies, brand-names of cars, etc. – repeat through different domains of business applications. If this is true, then we can program a collection of reusable entity-specific text-parsing rules and reapply them wherever possible. Currently there is not enough empirical evidence to confirm or deny this hypothesis.

3.2 Data Index

It would take too long time to match all the relevant data instances in the target database to the user question. With the help of the data index, which is similar to inverted index or inverted file in Information Retrieval, the system quickly selects fewer candidates for closer inspection. A candidate is a data instance whose text contains at least one word represented in the user question.

3.2.1 Structure of the Data Index

The logical structure of the data index is shown in Figure 3.4.

Ordered array of truncated words of all the data instances from all the entities facilitates access to data instances that contain a particular word. Words longer than four letters may be truncated, which allows matching of plural-singular forms of nouns, different tenses of verbs, related verbs and nouns (see Section 3.2.2). A truncated word, which ends with an asterisk, often happens to be the word's stem. For the sake of simplicity, truncated words are called stems further in this thesis.

Arrays of identifiers pointing to data instances, attached to each stem, links the stem to data instances that contain at least one word having this stem.

We may observe that the logical structure of the data index resembles that of a hash table [Cormen et al. 1990, pp. 221-226].

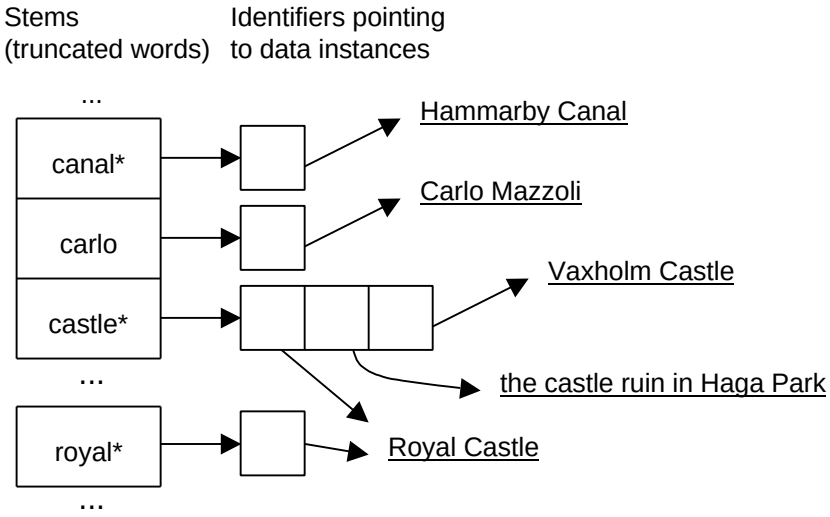

Figure 3.4 Logical structure of the data index.

The physical structure of the data index is different. SQL determines its table-like shape. Table 3.1 shows the columns and sample data in the data index.

Table 3.1 Physical structure of the data index

Word	Stem	Entity name	Primary key
canal	canal*	place	523
carlo	carlo	performer_artist	319
castles	castle*	place	19
castle	castle*	place	113
castle	castle*	place	181
cathedral	cathedral*	place	126

Word contains words as they appear in data instances that come from the textual data columns of the SQL queries of entity definitions (see Sections 3.1.1 and 3.1.2). The words are obtained after the necessary text-parsing rules or the "ruleless" algorithm are applied. This allows dropping unnecessary articles and prepositions and including split forms of the words (e.g., "Di Caprio" and "DiCaprio"). Further, the words in *Word* are used by the spelling checking facility to recover possible misspellings in user questions.

Stem contains the stems of the words in *Word* (more precisely, truncated words). The stems are assigned, and the *Stem* column filled, by the administration tool of the question assistant. This process is fully automated. If the tool does not know the stem, it copies the entire word from *Word* to *Stem*, as in the case of "carlo" in Table 3.1.

Entity name contains the names of the entities where the data instances belong.

Primary key contains the primary keys of the data instances. The values of the primary keys come from the columns called “pk” in the SQL queries of entity definitions (see Section 3.1.1).

A unique combination of an entity name and a primary key constitutes the global identifier of a data instance.

3.2.2 Stemming and Query Expansion

Stemming in Information Retrieval

In an Information Retrieval system, stemming is used to reduce variant word forms to common roots, and thereby improve the ability of the system to match query and document vocabulary. The variety in word forms comes from both inflectional and derivational morphology, and stemmers are usually designed to handle both. Stemming enhances recall (i.e., the ability to retrieve all relevant documents in the collection) because it expands the original query with related word forms. Stemming can improve precision at low recall levels by promoting relevant documents to high ranks [Xu and Croft 1998, pp. 61-62].

Generally, document retrieval using stemming is better than no stemming at all. [Hull 1996] reports average absolute improvement (supposedly, Hull means better recall and precision) due to stemming by 1-3%, but it makes a large difference for many individual queries. Stemming is more important for short queries and short documents in the collection. [Krovetz 1993] reports increase in precision by around 40% for queries with the average length of 7 words and documents with the average length of 45 words. [Popovic and Willett 1992] shows that stemming had increased recall by 30% for Slovene text using 2-15 words short queries. Slovene is morphologically more complicated than English: the stemmer removes no less than 5276 suffixes, whereas the Porter stemmer [Porter 1980] for English removes about 60 suffixes. [Carlberger et al. 2001] reports test results where that stemming had increased precision for Swedish text by 15%. The average length of the queries was 2.7 words, and the average length of the test documents was 181 words.

The question assistant indexes data in the target database – names and titles of real world objects – which are very short “documents”, usually less than 10 words long. User queries do not exceed one sentence. Often users submit only one keyword. Thus, the user queries and the “documents” processed by the question assistant are even shorter than those processed by Information Retrieval systems. Considering the fact that stemming proved especially valuable for short queries and short documents, it is even more important for the question assistant than for an average Information Retrieval system.

Irregular Forms of Words, Query Expansion

Stemming as a means of conflating words to a common root has a number of limitations:

- The question assistant does not permit stems shorter than four letters. For example, it does not truncate “puts”, “putting” into “put*” or “actor”, “actress” into “act*”.
- Irregular forms of verbs and nouns may have no common root, e.g., “see”, “saw” or “be”, “am”, “is”, “are”.
- Word stems may be too ambiguous, e.g., “color” cannot be stemmed as “color*” which matches “Colorado”.
- Both British and American English forms must be present, e.g., “litre” and “liter”, “theatre” and “theater”, “colour” and “color”.
- Switching between related nouns and verbs may bring up too ambiguous truncated forms. For example, “disprove” and “disproof” cannot be conflated to “dispro*”; “cello” and “cellist” cannot be conflated to “cell*”.
- Latin words have variants, e.g., “anaesthesia”, “anesthesia”.
- Stemming does not resolve common synonymy, e.g., “decipher” and “decode”, “crayfish” and “crawfish”.
- Stemming ignores links between alternative names and abbreviations, e.g., “Gothenburg” and “Göteborg”, “Bill Clinton” and “William Jefferson Clinton”, “Stockholm” and “Sthlm”.
- Because the question assistant is operated in Sweden, the questions tend to contain words with spelling influenced by the Swedish language, for example “Afrika” instead of “Africa”, “tabell” instead of “table”.

The above problems can be solved by query expansion. In Information Retrieval, query expansion is a method of adding new search keys (i.e., words and phrases, and the operators that express requirements for their occurrence in the text) to obtain a better correspondence between the query and documents carrying potential information. Expansion keys are usually elicited from the search results of an original (unexpanded) query or some external source, such as vocabularies [Kekäläinen 1999, p. 3].

The question assistant adds synonyms to the words in the original user question. The synonyms come form a manually created repository of irregular forms. The repository contains clusters of equivalent words or stems, such as:

- “am”, “are”, “be”, “been”, “being”, “is”, “was”, “were”;
- “actor*”, “actress*”, “artist*”, “performer*”;
- “boxer*”, “boxing*”.

A cluster may contain equivalents of a word in another language, or frequent misspellings. For example, Swedish people may write “museet” when they refer to a museum in Stockholm.

When the question assistant receives a user question, it expands the question: if a word has one or several corresponding clusters in the repository of irregular forms, the word gets a number of synonyms. Figure 3.5 demonstrates an example.

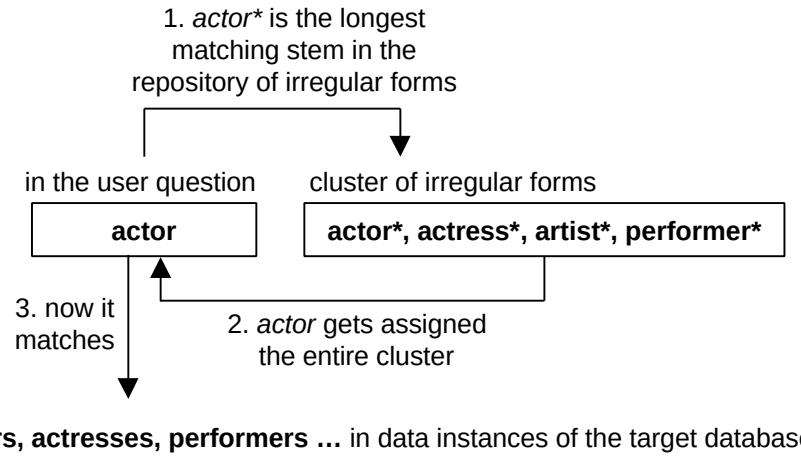

Figure 3.5 Example of assigning a cluster of equivalent stems.

According to the above figure, the word “actor” in the user question is assigned the stems “actor*”, “actress*”, “artist*”, and “performer*”. Since now, “actor” matches a number of equivalent words, such as “actors”, “actresses”, “performers”, etc.

The clusters experience the old problem of NLP lexicons – word sense ambiguity. “Being” has different meanings in “is being” and “a human being”. “Boxers” may refer to people engaged in the sport of boxing, two dogs, or men’s underwear. To make things worse, the clusters may overlap. For example, “lay” belongs to both (i) “lie”, “lies”, “lay”, “lain”, “lying” and (ii) “lay”, “lays”, “laid”, “laying”.

The repository of irregular forms is used exclusively to enhance the recall of the retrieval of data instances. Despite pessimistic expectations, ambiguous synonymy in the clusters does not much undermine the performance of the question assistant. The are reasons for that. (i) First, a data instance, which typically identifies a real world object, is a specific combination of words. For example, while the words “stock” and “exchange” are ambiguous, “stock exchange” turns out a different concept less affected by the ambiguity of its constituents. (ii) Further, data instances contain mostly nouns, therefore the processing is less affected by ambiguous verbs and adjectives. (iii) Finally, a user is expected to ask specific questions in order to get specific answers. Ambiguities in the question permit ambiguities in the answer.

The repository of irregular forms poses a number of questions to its developer:

- If we include ordinary language synonyms in a cluster, how deep synonymy do we consider?

- Similarly, if we switch between related nouns, verbs, and adjectives, how deep relations do we consider?
- Where lies the optimal edge between introducing knowledge-domain-specific synonymy in the repository of irregular forms and data-instance-specific synonymy in x-rules? How do we optimize the trade-offs?
- Would the performance of the system be worse if we use an ordinary lexicon, such as WordNet[33]? [Hull 1996] reports that there are problems with using linguistic approaches for stemming in Information Retrieval. Many decisions about the root form of a word, which are properly motivated from the linguistic perspective, are not optimal for Information retrieval performance. Linguistic analysis tools must be tailored for Information Retrieval applications and preferably optimized for the particular domain. On the other hand, the question assistant is not an Information Retrieval application (i.e., it does not query a collection of free text documents but rather answers questions using information in a relational database) although it uses quite a few Information Retrieval techniques.

This thesis does not answer the above questions.

Manual vs. Automated Stemming

In Information Retrieval, stemming can be done using stem dictionaries or suffix removal algorithms. [Hull 1996] argues that prefix removal can be an undesirable feature. [Popovic and Willett 1992] observes that, for Slovene text, there is no significant difference in the effectiveness of document retrieval that uses manual right-hand truncation (i.e., a manually created stem dictionary) and an automatic stemming algorithm.

Stem dictionaries and stemming algorithms both have their positive and negative aspects. A positive aspect of a stemming algorithm is that it does not require an exhaustive collection of stems. The algorithm is designed for the language in general. A negative aspect is that a stemming algorithm makes mistakes. The two most popular algorithms of suffix removal – [Lovins 1968] and [Porter 1980] – ignore the meanings of the words, which leads to a number of stemming errors. For example, the Porter stemmer conflates “general”, “generous”, “generation”, and “generic” to the same root [Hull 1996].

A stem dictionary is more reliable because the stems are created using intelligent human reasoning over the meanings of the words. A negative aspect of a stem dictionary is that it requires an exhaustive selection of stems. The dictionary cannot deal with words if it contains no corresponding stems.

---

[33] http://www.cogsci.princeton.edu/~wn/, valid in November 2001

Both a stem dictionary and a stemming algorithm profit from selecting the longest matching stem. Having two options – "admir*" and "admiral*" – "admirer" matches the first one but "admirals" match the second one.

Another utility of an exhaustive stem dictionary is being a collection of standard words used in order to detect misspellings: if a word in a user question has a representative stem, then the word is not misspelled and spelling correction should not take place.

The question assistant uses a stem dictionary with about 9500 stems in it. Both common English words and unusual words from the SIS database are included.

Assigning Stems and Irregular Forms of Words

Matching of a word in a data instance to a word in a user question uses a common stem, if such a stem is available. It does not matter whether the stem is first assigned to the word in a user question or the word in a data instance. The question assistant assigns stems to words in data instances in the target database when it indexes these data instances. [Xu and Croft 1998] describes stemming at the indexing time as computationally more efficient. Besides, stems in the data index enable a visual control over the stem-word links, which helps to detect cases where wrong stems are assigned.

On the contrary, query expansion works with words in user questions. The question assistant extends each word in a user question into a class of equivalent words, if there are appropriate word clusters in the repository of irregular forms. Matching to a word in the user question takes matching to each word in the class of equivalent words.

3.2.3 Spelling Correction

If a word in a user question does not match any stem in the data index, the stem dictionary, the repository of irregular forms, the stop-list, keywords in question templates (the stop-list and keywords in question templates are discussed in Section 3.3.4), the word is considered misspelled. This, in fact, may be a false assumption based on the limited vocabulary of the question assistant. Spelling correction is done with respect to the *Word* column (Table 3.1): the system selects the words that have spelling similar to that of the misspelled word. The newly selected words are further treated as irregular forms of the misspelled word.

Because spelling correction is not in the focus of this research, the question assistant has no good spelling correction algorithm. Instead, the system uses the "difference($string_1$, $string_2$)" function of the Microsoft SQL Server. The function provides a good input for a more rigorous spelling correction algorithm, but it is not that good if used instead of such an algorithm.

3.3 Question Templates

Question templates are the foundation of this question-answering approach. They are very similar to the FAQ entries described in Parts 2 and 3 of this thesis. The only significant innovation is entity slots (see Section 1.1), which makes it possible to tailor the "frequently asked questions" and include data instances from the target database. Figure 3.6 shows a part of the form for editing a question template (the erased part is for editing the corresponding answer template).

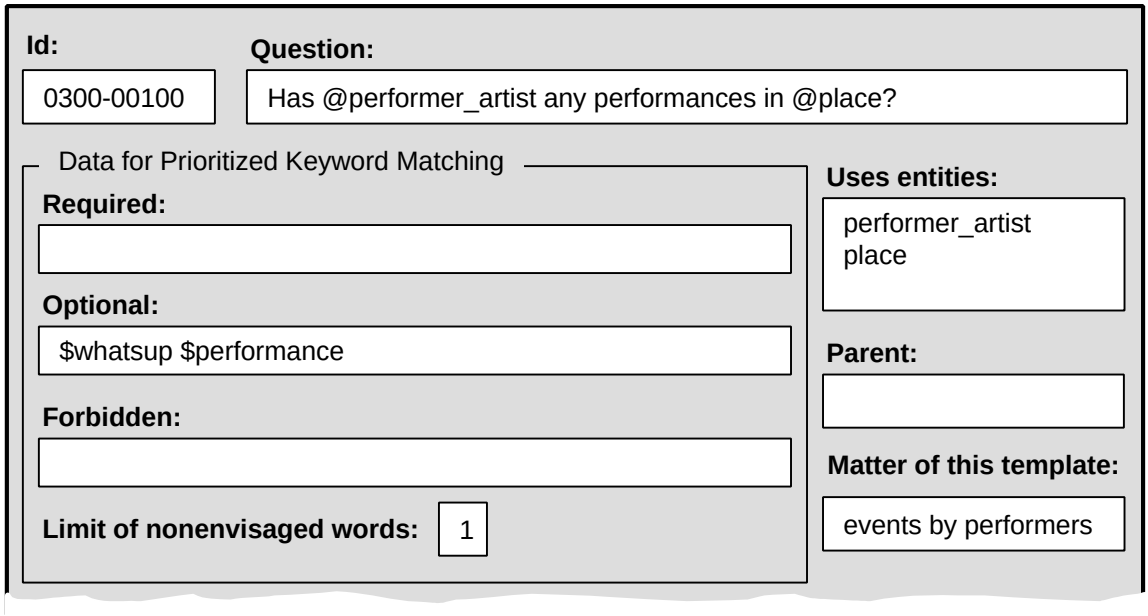

Figure 3.6 Sample question template in the edit form.

3.3.1 Entities Used in a Question Template

In Process 1 (Figure 2.6), the question assistant finds out which data instances are referenced in the submitted user question. When moving to Process 2, which deals with question templates, the question assistant knows which entities are referenced in the user question, as each data instance belongs to some entity.

Normally a question template uses 1-2 entities, but there is no formal limit. There is an explicit list of all the entities used by the question template and the associated answer template ("uses entities" in Figure 3.6).

When the system matches a user question to a question template, first it verifies whether or not the user question references all the entities used by the question template. For example, the user question "When does Termini sing in Oscar's Church?", where Termini belongs to "performer_artist" and Oscar's Church belongs to "place", would match the sample question template in Figure 3.6 because the user question references both entities used by the template. On the contrary, "When does Termini perform?" would not match the template because this question does not reference the entity "place".

A question template with no entity slots is an ordinary static FAQ entry. The question assistant processes FAQ entries the same way as the FAQ answering systems, discussed in Parts 2 and 3 of this thesis, do. In this part of the thesis we consider only the question and answer templates that have entity slots.

3.3.2 Question Templates as Predicates

Entity slots introduce another view on a question template – it represents a predicate. Let us consider the template in Figure 3.6 "Has `@performer_artist` any performances in `@place`?" In case if the answer is "yes", we would like to find out the time of the performances. Hereby, the question template is the predicate in the logical statement

∃ `@time`: `@performer_artist` performs in `@place` at `@time`

which has a sequence of variable and fixed parameters: variable parameter `@time`, fixed parameters `@performer_artist` and `@place`. During the question-answering process, the values of fixed parameters are bound to the user query whereas the values of variable parameters are to be found. For example, when the system matches the above template to the user question "Does Termini sing in Oscar's Church?", it tries to find the set of all `@time` values that satisfy the condition of the predicate:

{ `@time` | `@performer_artist` performs in `@place` at `time` &
`@performer_artist` = "Termini" & `@place` = "Oscar's Church" }

3.3.3 The Question

The question of a question template is a dynamic FAQ. The text of the question contains entity slots – words that begin with "@" – that identify the key concepts of the question and can be replaced with data instances that belong to the corresponding entities. Each entity slot is declared in the list of the entities used by the template ("uses entities" in Figure 3.6).

In Process 3 (Figure 2.6), the question assistant transforms matching question templates into interpretations of the user question and replaces entity slots with data instances referenced in the user question (example in Figure 1.3). If a reference to data instances in the user question is ambiguous, the question assistant offers a drop-down list of data instances that match the ambiguous reference (Figure 2.4). The ambiguity is relative with respect to the collection of data instances in the target database.

According to an entity definition, a data instance has two components: its primary key and its textual data (see Section 3.1.2). It is the *textual data* that fills the entity slots in a question.

3.3.4 Attributes for Prioritized Keyword Matching

After the question assistant has selected the question templates that use only the entities relevant for the submitted user question, the system refines the selection by means of Prioritized Keyword Matching the same way as the FAQ answering systems, discussed in Parts 2 and 3 of this thesis, do.

The Keywords

Each question template has the required, optional, and forbidden keywords that distinguish its question (see the chapter “Prioritized Keyword Matching” in the paper “Automated FAQ Answering: Continued Experience with Shallow Language Understanding” in Part 3 of this thesis).

Required keywords are words that convey the essence of the question. A question template matches a user question only if all the required keywords have counterparts in the user question.

Entity slots have slightly modified the initial meaning of required keywords coming from FAQ entries. Because entity slots identify the key concepts involved in the question, they act required keywords. Yet there is no need to indicate entity slots among required keywords because each question template has an explicit list of the entities being used. As we see in Figure 3.6, the template has no required keywords. In fact, the entities “performer_artist” and “place” represent two required keywords.

The new main roll of required keywords is to indicate relationships and relevant attributes of the key concepts represented by entity slots. In Figure 3.6, no relationships are explicitly indicated. Within the context of the SIS database, there is only one relationship between an artist and a place: the artist performs in the place. For the sake of simplicity this only relationship is not specified.

Optional keywords help to convey the meaning of the question but can be omitted without changing its essence. The nuances may change though. By matching both required and optional keywords, the system makes subtler conclusion about the closeness between the question template and the user question.

Forbidden keywords (also called negative keywords) indicate the concepts that are not compatible with the meaning of the question in a template. We indicate forbidden keywords only if we need to emphasize the difference between two similar but still not equal question templates in the repository. If a forbidden keyword has a counterpart in the user question, the match between the template and the question is rejected.

The system has also a list of generally “irrelevant” words, such as “a”, “the”, “is”, etc., known as *stop-words* in Information Retrieval. Stop-words are optional keywords common for all the question templates.

The user question matches a question template if the required and optional keywords match the majority of the words in the question, given that the stop-words in the question are ignored. How much is "the majority"? There is a *limit of nonenvisaged words* (the words in the user question that do not match required or optional keywords) individual for each template. The limit is usually 0 or 1.

Each keyword is represented by a number of synonyms. A synonym may be a single word or a phrase (see the chapter "Phrases" in the paper "Automated FAQ Answering: Continued Experience with Shallow Language Understanding" in Part 3 of this thesis; there are no phrases in Part 2). A synonym that starts with "$" is a *substitute* which represents an often reused group of synonyms kept in the repository of substitutes (see the chapter "Substitutes" in the paper "Automated FAQ Answering: Continued Experience with Shallow Language Understanding" in Part 3 of this thesis). In the example in Figure 3.6, "$whatsup $performance" are references to two separate groups of synonyms held in the repository of substitutes.

When matching a synonym of a keyword to the user question, the question assistant ignores query expansion (see Section 3.2.2). Query expansion is used only for matching data instances in the target database. The administrator of the system keeps control over context sensitive synonyms by writing them directly into the edit fields for the keywords or by choosing appropriate substitutes. This increases the precision of matching.

No Entity Slots among the Keywords

No entity slots are permitted among required, optional, and forbidden keywords. As it was previously discussed, the list of the entities used by the question template ("uses entities" in Figure 3.6) specifies entities that act as required keywords. Theoretically, it may become necessary to specify phrases that contain entity slots, for example, "<in; `@place`>" or "<`@place` # `@city`>". So far there has been no need for that.

Close, Related, and Complementary Interpretations of the User Question

In Parts 2 and 3, the Prioritized Keyword Matching algorithm distinguishes FAQs that are close or just related to the user question. In Process 3 (Figure 2.6), where the question assistant creates interpretations of the submitted user question, the system distinguishes three grades of proximity of a question template to the user question: a close, related, or complementary interpretation.

C*lose interpretation* uses all the entities that the user question refers to, all the required keywords and no forbidden keywords match the user question, the number of the words in the user question that do not match required and optional keywords *does not* exceed the limit of nonenvisaged words for this template. In Figure 1.3, a close interpretation follows the heading "Question assistant understood your question as".

Related interpretation is similar to the close interpretation except the number of the words in the user question that do not match required and optional keywords *does* exceed the limit of nonenvisaged words for this template. In Figure 1.3, a related interpretation follows the heading “Answer to the following question may be interesting too”.

Complementary interpretation uses all the entities that the user question refers to, no forbidden keywords match the user question, and there may be required keywords that do not match the user question.

The above classification of interpretations of the submitted user question minimizes the number of “sorry, no answer found” replies because the question assistant has almost always something to offer. Nonetheless, the first testing has shown that complementary interpretations may be confusing and make the user think that the system has not “understood” the user question. Complementary interpretations are likely to be replaced by something else in the future.

3.3.5 Auxiliary Templates

Auxiliary question templates are equivalent to auxiliary FAQ entries introduced in Section 3.2.1 of Part 2 of this thesis.

One can formulate the same question differently. Although the approach with required and optional keywords is rather flexible, sometimes one question template cannot represent all the possible forms of the corresponding questions. Therefore there are clusters of question templates: a number of equivalent question templates have the same answer template. All templates in a cluster use the same entities. “Parent” in Figure 3.6 contains the identifier of the main question template of the cluster which has the associated answer template. No parent means that this is the main template of the cluster.

All auxiliary question templates in a cluster represent the same predicate (see Section 3.3.2).

3.3.6 ISA and PartOf Relationships

In everyday language, we use synonymy (e.g., “phone” is a synonym of “telephone”), generalization and specialization (e.g., a mobile phone is a phone), the whole-part and part-whole relationship (e.g., a car has an engine, and the engine is a part of the car). In conceptual modeling, ISA (“is a”) and PartOf are conventional relationships between concepts. The question assistant perceives these relationships as (i) synonymy or (ii) context sensitive synonymy. Synonymy and context sensitive synonymy between entities are handled by means of ordinary entity definitions and question templates. The system has no specific features implemented to deal with them.

Synonymy

Establishing synonymy between several entities is an easy task if they are implemented in the same data column. Then we define a parent entity. Typically, data instances that belong to different child entities are distinguished by a label attached to the name (e.g., "select the names of the performers whose role is 'dancer'") or the relationships to other entities (e.g., "select the names of the performers who take part in the events that fall under category 'music'"). By manipulating with the labels and relationships in the SQL statement that defines the parent entity, we can establish synonymy between child entities as needed.

Establishing synonymy between several entities is a more difficult task if the entities are implemented in different data columns. The SQL "union" operator would be appropriate to join several columns, but it is not allowed in order to avoid duplicate primary keys. An alternative solution is introduction of equivalent question templates for each synonym entity separately.

Context Sensitive Synonymy

Context sensitive synonymy is more complicated. Let us consider the question "When does U2 come to Globen?" Because Globen is located in Stockholm (i.e., Globen is a part of Stockholm) and the band arrives from abroad, a more common question would be "When does U2 come to Stockholm?" In this context Globen can be exchanged with Stockholm, in some others cannot. "How many seats are there in Globen?" is an intelligent question, whereas "How many seats are there in Stockholm?" is not.

The question assistant handles context sensitive synonymy by having separate, equivalent but still different, question templates for both entities involved in the ISA or PartOf relationship. The equivalent templates are created only within the context where synonymy is appropriate. This approach may become tedious if there are long chains of repeated ISA or PartOf relationships that cannot be treated as ordinary synonymy. So far it has not been the case.

3.3.7 Matter Strings

Each question template has a matter string – a two-three words long description of the essence of the question in the template – such as "event categories", "activities held at designated places", "ticket and price information", etc. The only purpose of matter strings is to create an illusion that the question assistant is more intelligent than it actually is.

Let us consider an example of asking a question in Figure 3.7. The user has asked "Globen next week?" Because the question was vague, the question assistant had no accurate answer. Instead, the system retrieved some information it had about the location called Globen (in fact, "Globen" is also an ambiguous name, therefore the system offered several its interpretations). In the end, the question assistant printed

the following: “Question assistant ran also queries concerning performers at designated places with respect to your question, but no appropriate answers were found.” The piece of text “performers at designated places” is the matter string of the question template “Who has a performance in/at `@place` during the period `@period`?”

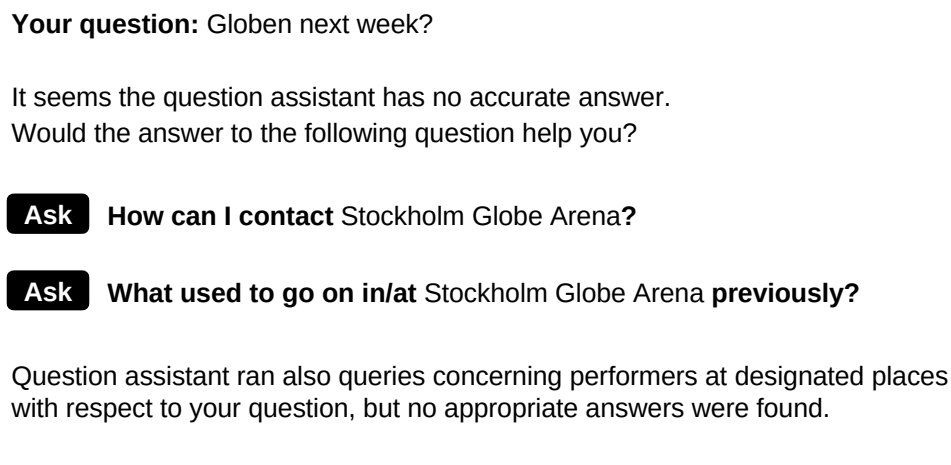

Figure 3.7 Example of a footnote with an included matter string (the two bottom lines).

By printing a footnote with an included matter string, the question assistant showed that it “understood” that Globen was a location for public performances, it queried the database for performances at Globen, but no performances were registered for the period “next week”.

3.3.8 What Is a Good Question Template?

A well-answered question is one where the question assistant has managed (i) to recognize relevant data instances and ignore irrelevant data instances, and (ii) to match relevant question templates and ignore irrelevant question templates. A good question template is one which *does* match a large variety of differently formulated user questions with the meaning close to that of the question in the template, and *does not* match not related user questions. Good question templates have the same characteristics as FAQ entries discussed in Parts 2 and 3 of this thesis, with some modifications.

Sufficient selection of auxiliary templates helps to meet a broad range of formulations of a user question.

Sufficient selection of the templates that implement context sensitive synonymy between entities (i.e., the ISA and PartOf relationships) helps to deal with generalizations and specializations of data instances.

Good selection of attributes for Prioritized Keyword Matching ensures the quality of matching the template to a user question, which implies (i) thorough selection of

required and optional keywords, and (ii) thorough selection of words and their stems as synonyms of each keyword.

3.4 Answer Templates

In Process 4 (Figure 2.6), the user selects an interpretation of his or her original question (example in Figure 1.3) and passes it to Process 5. The actual data passed is the identifier of the selected question template and the primary keys of the pertinent / selected data instances. In Process 5, the question assistant retrieves the answer template associated with the indicated question template and processes it with respect to the data instances. The answer template creates the answer of the user-selected interpretation. It answers also the initial user question if the system has managed to find a right interpretation of the question.

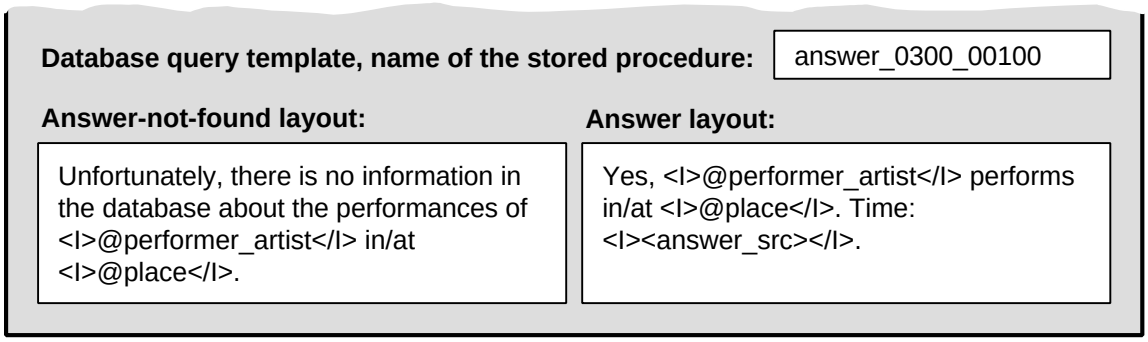

Figure 3.8 Sample answer template in the edit form.

Figure 3.8 shows a part of the form for editing an answer template. The sample answer template is associated with the question template in Figure 3.6. An answer template comprises a database query template, an answer layout template, and an answer-not-found layout template.

3.4.1 Database Query Templates

A database query template is a stored procedure on a Microsoft SQL Server. The identifier “answer_0300_00100” in Figure 3.8 is the name of the stored procedure:

```
CREATE PROCEDURE answer_0300_00100
@performer_artist INTEGER,
@place INTEGER
AS
SELECT start FROM timetable
WHERE timetable.eventid IN
( SELECT eventid FROM event
  WHERE event.placeid = @place
  AND event.eventid IN
```

```
    ( SELECT eventid FROM performer
      WHERE performer.performerid = @performer_artist
    )
  ) ORDER BY start
```

The input parameters *@performer_artist* and *@place* are entity slots, they are the *fixed parameters* in the predicate (see Section 3.3.2) represented by the question template. The stored procedure queries the target database after the necessary values are passed in its input parameters. The names of the input parameters aka entity slots are equal to the names of the corresponding entities and are declared in the list of the entities used by this answer template and its corresponding question template ("uses entities" in Figure 3.6).

According to an entity definition, a data instance has two components: its primary key and its textual data (see Section 3.1.2). These are the *primary keys* that fill the entity slots in a database query template.

A database query template returns data in a number of columns of any data type. The data columns are the *variable parameters* in the predicate represented by the question template.

The answer and answer-not-found layout templates utilize the data retrieved by the database query template.

3.4.2 Answer Layout Templates

An answer layout template is a piece of HTML text which contains, besides ordinary HTML tags, entity slots as well as the question-assistant-specific `<answer_src>` and `<answer_clause>` tags (see the example in Figure 3.8).

After the question assistant has processed the database query template and retrieved some data, it processes the answer layout template. The system:

1. replaces the entity slots in the answer layout template (e.g., `@performer_artist` and `@place`) with the textual data of the pertinent data instances;
2. processes the `<answer_src>` and `<answer_clause>` tags and includes formatted text of the data instances retrieved by the database query template (e.g., formatted text of `start` retrieved by `SELECT start FROM timetable WHERE ...`);
3. submits the resulting HTML document back to the user's Web browser.

In Figure 3.9, we see that the answer has two parts: the text of the user-selected interpretation of the initial question (in bold) and the text of the answer itself, created by the answer template, which answers the user-selected interpretation.

Has Claudia Termini any performances in Oscar's Church?

Yes, *Claudia Termini* performs in/at *Oscar's Church*. Time: *2000-08-09 20:00*.

Figure 3.9 Answer of a user-selected interpretation.

The <answer_src> tag

This tag embodies the data instances retrieved by the database query templates. It has three formats: a table, a list, and a single item.

Table. The data instances are to be printed in a table. By using the tag, we define the place for the table in the HTML text, the number of columns, and the headings of the columns. For example, `<answer_src | Time | Event>` tells the question assistant that the data instances are to be printed in two columns with headings "Time" and "Event". The number of columns defined in the `<answer_src>` tag is equal to the number of columns retrieved by the database query template. The following answer layout template has produced the answer in Figure 3.10:

```
The following events take place in/at <I>@place</I> during
the period <I>@period</I>.<P>
<answer_src | Time | Event>
```

What goes on in/at Skansen during the period 9 June 2001 - 13 June 2001?

The following events take place in/at *Skansen* during the period *9 June 2001 - 13 June 2001*.

Time	Event
2001-06-09	Sting The world-renowned performer comes to Stockholm.
2001-06-10	Starblast 2001, talent show finals The finals in the Starblast 2001 (Stjärnskott 2001) talent competition take place on the Sollidenscenen stage. Only ten finalists are left of the hundreds who started out. Come and see the stars of tomorrow.
2001-06-13 18:00	Music on a Summer Evening, concert Concert in the Seglora Church in Skansen.

Figure 3.10 Answer of a user-selected interpretation, data instances printed in a table.

List. The tag without any parameters tells the question assistant that the data instances are to be printed in a comma separated list. In Figure 3.9, where the answer is produced by the answer layout template shown in Figure 3.8, the time "2000-08-09 20:00" is the only item in the list.

Single item. A number as the only parameter indicates the number of the column (counting starts from 0) whose first data instance is references. For example, `<answer_src 1>` references the first item in the second (0^{th}, 1^{st}) column retrieved by the database query template. The single item tags are used together with the `<answer_clause>` tag.

The `<answer_clause>` tag

The following answer layout template has produced the answer in Figure 3.11:

```
Contact <I>@place</I> as below:<UL>
<answer_clause 0>
    <LI>address: <answer_src 0>
    <answer_clause 1>, <answer_src 1></answer_clause>
</answer_clause>
<answer_clause 2>
    <LI>public phone number: <answer_src 2>
</answer_clause>
<answer_clause 3>
    <LI>e-mail address: <answer_src 3>
</answer_clause>
<answer_clause 4>
    <LI>URL: <answer_src 4>
</answer_clause>
</UL>
```

How can I contact Åhléns City?

Contact *Åhléns City* as below:

- address: Klarabergsgatan 50, Stockholm
- public phone number: 08 - 676 60 00

Figure 3.11 Answer of a user-selected interpretation, specific data clauses printed.

As we see in the answer layout template, the address, public phone number, e-mail address, and URL of Åhléns City had to be printed. Nonetheless, only the address and the phone number were actually printed because the other data was not present in the target database. The unnecessary captions "e-mail address:" and "URL:" were dropped.

The pair <answer_clause></answer_clause> sets the boundaries for a piece of text that surrounds a data instance (<answer_src> tag) and is dropped if the data instance has the NULL value. The <answer_clause></answer_clause> tags can be embedded.

3.4.3 Answer-Not-Found Layout Templates

If in Process 4 (Figure 2.6) the user selects an interpretation with no answer, he or she gets "Unfortunately, there is no information in the database…". The exact text of the message is created by the answer-not-found layout template.

An answer-not-found template contains HTML text and entity slots. There are no <answer_src> and <answer_clause> tags because there are no data instances.

3.5 Transaction Logs

The dialog between a user and the system is recorded in order to find out the users' needs and to monitor the system's performance. A log contains the following information:

Query

- date and time of the transaction;
- IP address of the user's computer;
- numeric code of the action and its result:
  - 1 – interpretation selected (in Process 4, Figure 2.6),
  - 0 – question asked and answered – semantically close interpretations found,
  - -1 – question asked but not answered,
  - -2 – question asked, no semantically close interpretations found, yet still related or complementary interpretations were created,
  - -9 – abnormal termination because of a programming error.
- text of the user question if the action code is not 1, or the identifier of the question template of the selected interpretation otherwise;

System's Reply

- optionally in a new line, an error message;
- optionally in a new line, information about semantically close interpretations – semicolon-separated list of the following:
  - identifier of the question template,
  - comma-separated list of the following:

  - name of the entity,
  - list of the primary keys of the pertinent data instances.
- optionally in a new line, information about the related question templates and matching data instances, format the same as in the previous item.

The logs are stored in a structured ASCII file and separated by an empty line. The new line character, semicolon, and comma within the stored data items are encoded by a hexadecimal number. An example of a log:

```
01/07/2001 16:08:25;130.237.157.144;-2;Globen next week?
RL:0200-00200,place 1934 2536 1603;0200-00104,place 2536
1603;0200-00300,place 1934 2536 1603
```

It is not easy to understand what is written in a log. Therefore the administration tool of the question assistant offers a converter. After conversion the above log looks like this:

01/07/2001 16:08:25; l412.dsv.su.se; related or complementary only (-2); **Globen next week?**
Related interpretations:
0200-00200 How can I contact @place? | *@place* "glob"; "Stockholm Globe Arena"; "Annexet, Stockholm Globe Arena"
0200-00104 What used to go on in/at @place previously? | *@place* "Stockholm Globe Arena"; "Annexet, Stockholm Globe Arena"
0200-00300 What other places are in walking distance from @place? | *@place* "glob"; "Stockholm Globe Arena"; "Annexet, Stockholm Globe Arena"

3.6 Summary

Entity definitions, the data index, question templates, answer templates, and transaction logs are all but one components of the question assistant. The question processor – the main actor of the system – is left outside this list.

Entity definitions establish the links between entity slots and the pertinent data instances. Each entity definition has a name which is the name of the entity and the corresponding entity slots. An SQL query, one for each entity definition, defines which data instances belong to the given entity and may fill the corresponding entity slots. A data instance has two components – its primary key and its text, which usually is a name of a real world object. The primary keys are used in database query templates, in other cases the text is used.

When the question assistant matches the text of a data instance to a user question, it considers a number of synonyms of this text. In order to find out the synonyms, the system applies an appropriate text-parsing rule to the data instance. There are entity-specific and data-instance-specific text-parsing rules.

Data index helps the system to quickly select few – much fewer than in the entire database – data instances for matching to the user question. In the data index, words from the data instances are truncated, which allows matching of plural-singular forms of nouns, different tenses of verbs, related verbs and nouns.

Query expansion helps to resolve synonymy that cannot be resolved by the means of word truncation. Using the repository of irregular forms, the question assistant extends each word in a user question into a class of equivalent words, if there are appropriate irregular forms. Matching to a word in the user question takes matching to each word in the class of equivalent words.

Question templates are the foundation of this question-answering approach. They are very similar to the FAQ entries described in Parts 2 and 3 of this thesis. The only significant innovation is entity slots, which makes it possible to tailor the "FAQ" and include the data instances from the target database into the text of the "frequently asked question" and its answer.

Answer template creates the answer of the question expressed in the corresponding question template. The main component of an answer template is its database query template – a database query with entity slots. After the question assistant has processed the database query template and retrieved some data, it incorporates the data into the answer layout and sends the result – an HTML document – back to the user's Web-browser.

Transaction logs record the dialogs between users and the question assistant, which helps to find out the users' needs and monitor the system's performance.

4 Description of the Algorithms

For this research, most interesting are Processes 1-3 in Figure 2.6 – matching of data instances and question templates to a user question, and creating of interpretations of the user question. This section discusses the algorithms in Processes 1-3 and their estimated execution time.

4.1 Matching of Data Instances

The objectives of Process 1 are to find the data instances and their corresponding entities referred to in the user question.

4.1.1 Matching Process

The matching process has three parts. First, the system expands the query – the submitted user question – with irregular forms of the words. Second, the system uses the data index and locates candidate data instances. Third, the system inspects each candidate, and accepts or rejects it. The routine MATCHDATAINSTANCES, which implements Process 1, embraces all three parts and returns the data instances that match the user question.

The syntax of the pseudo-code is the same as in [Russell and Norvig 1995].

```
function MATCHDATAINSTANCES(question,
                            irregular,
                            index,
                            database,
                            rules)
  returns
    1) a table of data instances that match the user
       question, indexed by entity names:
```

Entity name	**Data instances grouped by entities where they belong to**			
entity name$_1$	data instance$_{11}$	data instance$_{12}$	…	data instance$_{1u}$
entity name$_2$	data instance$_{21}$	data instance$_{22}$	…	data instance$_{2v}$
…	…	…	…	…
entity name$_m$	data instance$_{m1}$	data instance$_{m2}$	…	data instance$_{mw}$

```
       where m ← number of entities referred to in question
             u ← number of data instances that match
                 question and belong to entity1
```

v ← number of data instances that match question and belong to entity$_2$
w ← number of data instances that match question and belong to entity$_m$

2) a table of the names of entities referred to in *question*, indexed by the words in *question*:

Words	**Entity names; data instances in these entities contain the given word**			
word$_1$	entity name$_{11}$	entity name$_{12}$	…	entity name$_{1x}$
word$_2$	entity name$_{21}$	entity name$_{22}$	…	entity name$_{2y}$
…	…	…	…	…
word$_n$	entity name$_{n1}$	entity name$_{n2}$	…	entity name$_{nz}$

where *n* ← number of words in *question*
x ← number of entities where at least one matching data instance contains word$_1$
y ← number of entities where at least one matching data instance contains word$_2$
z ← number of entities where at least one matching data instance contains word$_n$

/* *the table keeps record of which entities are* */
/* *referred to by a given word in* *question* */

inputs
question, the user question
irregular, an array of clusters of irregular form of words - the repository of irregular forms
index, the data index
database, the data instances
rules, text-parsing rules

locals
expanded_query, an array of terms, where a term is an array of equivalent words/stems
candidates, an array of identifiers of data instances
data_instances, the table for returns (1)
entities, the table for returns (2)

01 *expanded_query* ← EXPANDQUERY(*question*, *irregular*)
02 *candidates* ← SELECTCANDIDATES(*expanded_query*, *index*)

```
03  (data_instances, entities) ← INSPECTCANDIDATES(
        expanded_query, candidates, database, rules)
04  return (data_instances, entities)

function EXPANDQUERY(question, irregular)
/* see Section 3.2.2 about query expansion */
  returns
    expanded user query – an array of terms, where a term
    is an array of equivalent words/stems
  inputs
    question, the user question
    irregular, the repository of irregular forms – an array
        of clusters of irregular form of words
  locals
    words, an array of the words in question
    expanded_query, an array of terms, where a term is an
        array of equivalent words/stems
    irregular_clusters, an array of clusters of irregular
        forms of words
05  words ← split question into words
06  for i ← 1 to ARRAYLENGTH(words)
07    irregular_clusters ← select clusters from irregular
         where at least one irregular form matches words[i]
08    expanded_query[i] ← words[i]
09    for j ← 1 to ARRAYLENGTH(irregular_clusters)
10      tmp ← split irregular_clusters[j] into words/stems
11      expanded_query[i] ← expanded_query[i] ∪ tmp
12    end
13  end
14  return expanded_query

function SELECTCANDIDATES(expanded_query, index)
  returns
    an array of the identifiers of data instances
  inputs
    expanded_query, an array of terms, where a term is an
        array of equivalent words/stems
    index, the data index
  locals
    candidates, an array of identifiers of data instances
```

```
15  for i ← 1 to ARRAYLENGTH(expanded_query)
16    tmp ← select identifiers of data instances from index
            where index.stem matches expanded_query[i]
17    candidates ← candidates ∪ tmp
18  end
19  return candidates

function INSPECTCANDIDATES(expanded_query,
                           candidates,
                           database,
                           rules)
  returns
    1) the table for returns (1) in MATCHDATAINSTANCES
    2) the table for returns (2) in MATCHDATAINSTANCES
  inputs
    expanded_query, an array of terms, where a term is an
        array of equivalent words
    candidates, an array of identifiers of data instances
    database, the data instances
    rules, text-parsing rules
  locals
    instance, a data instance
    rule, one of the rules in rules
    synonyms, an array of the synonyms of instance
    candidate_ok, "true" of "false"
    entity, an entity name
    word, a word
    data_instances, the table for returns (1)
    entities, the table for returns (2)
20  for i ← 1 to ARRAYLENGTH(candidates) do
21    instance ← select the data instance from database
                 where the identifier is candidates[i]
22    rule ← select a rule from rules for instance, or
            the "ruleless" algorithm if there is no rule
23    synonyms ← CREATESYNONYMS(instance, rule)
24    candidate_ok ← "false"
25    clear the marks in expanded_query to be made again in
      line 27
```

```
26    for j ← 1 to ArrayLength(synonyms) do
27      tmp ← match synonyms[j] to expanded_query,
              mark each matching term
28      candidate_ok ← candidate_ok or tmp
29    end
30    if candidate_ok then
31      entity ← the name of the instance's entity
32      data_instances[entity] ← data_instances[entity] +
            instance, index by entity name
            /* indexed for credibility test, see Section 4.3.1 */
33      for j ← 1 to ArrayLength(expanded_query) do
34        if expanded_query[j] is marked in line 27 then
35          word ← the original word in expanded_query[j],
                  the word as in the user question
36          entities[word] ← entities[word] + entity,
                             keep indexed by entity name
37        end
38      end
39    end
40  end
41  return (data_instances, entities)
```

The routine CreateSynonyms either (i) retrieves the x-rule associated with *instance*, if there is any x-rule, or (ii) creates a set of synonyms of the text of *instance* created according to *rule*. X-rules and the text of synonyms of *instance*, which both follow the syntax of keywords for Prioritized Keyword Matching, are interpreted by the routines that match data instances to a user question.

4.1.2 Speed of Data Access

The question assistant uses binary tree search which takes logarithmic time $c_a \log(n) + c_b$, where n is the number of data instances in the input [Cormen at. al 1990, p. 244].

The time of data retrieval by unique identifier depends on the database management system (DBMS). Ones that use pointers retrieve a data instance in constant time. An SQL server locates a primary key in logarithmic time. Let us introduce a DBMS-specific data retrieval time function dbms(n), where n is the number of data instances in the collection, as either constant or logarithmic:

$$\text{dbms}(n) = \begin{cases} C \\ c_a \log(n) + c_b \end{cases}$$

Let us introduce a *linear property* of the function: for any c_1, there exist c_2 and c_3 such that

$$\text{dbms}(c_1\, n) = c_2\, \text{dbms}(n) + c_3.$$

The property holds for both forms of dbms(n). If the function is constant, then $c_2 = 1$ and $c_3 = 0$. If the function is logarithmic, then $c_2 = 1$ and $c_3 = c_a \log(c_1)$.

4.1.3 Estimated Execution Time

The execution time of the matching process depends on the size of user questions, the size of data instances, and the number of data instances in the database. All user questions are one sentence long and contain a limited number of words. All data instances – mostly names of real world objects – have limited size. Hence, the execution time is a function T(N), where N is the number of data instances in the database.

EXPANDQUERY

The execution time of query expansion is limited by a constant because the sizes of the user question and the repository or irregular forms are limited by constants.

SELECTCANDIDATES

In line 16, `expanded_query[i]` is an array of words/stems limited by the maximum number of irregular forms c_2. How many identifiers (i.e., primary keys) are there selected into `tmp`? The number of identifiers associated with a given word/stem s is proportionate to the total number of data instance and the probability that a data instance contains s, which is $p(s)\,N$. For `expanded_query[i]` it makes $\sum_{i=1}^{c_1} p(s_i)\,N$, which is the number of identifiers in `tmp`. The time to find one identifier is less than $c_3 \log(N) + c_4$. Hence, the total execution time for line 16 is less than $\sum_{i=1}^{c_1} p(s_i)\,N\,(c_2 \log(N) + c_3) = c_2 \sum_{i=1}^{c_1} p(s_i)\,N \log(N) + c_3 \sum_{i=1}^{c_1} p(s_i)\,N$.

Line 17 merges two arrays, which results in a union of two sets. Before the system moves an item from `tmp` to `candidates`, it checks whether or not the item already exists in `candidates`. The number of items in `candidates` does not exceed the total number of items in `tmp` for all terms in `expanded_query`. The number of the terms is limited by c_4. Hence, the number of items in `candidates` does not exceed $c_4 \sum_{i=1}^{c_1} p(s_i)\,N$. The union of two sets takes time less than $\sum_{i=1}^{c_1} p(s_i)\,N \log(c_4 \sum_{j=1}^{c_1} p(s_j)\,N) + c_5$. Because $p(s_j)$ is a probability, $c_4 \sum_{j=1}^{c_1} p(s_j) \leq c_6$, and $\log(c_4 \sum_{j=1}^{c_1} p(s_j)\,N) \leq \log(c_6\,N) = c_7 + \log(N)$. Then he execution time for the union is less than $\sum_{i=1}^{c_1} p(s_i)\,N\,(c_7 + \log(\text{N})) + c_5 = \sum_{i=1}^{c_1} p(s_i)\,N \log(N) + c_7 \sum_{i=1}^{c_1} p(s_i)\,N + c_5$.

The total execution time for lines 16 and 17 is less than $c_2 \sum_{i=1}^{c_1} p(s_i)\,N \log(N) + c_8 \sum_{i=1}^{c_1} p(s_i)\,N + c_5$. Because the loop in lines 15-18 is executed c_5 times, the entire

routine SELECTCANDIDATES takes time less than $c_9 \sum_{i=1}^{c_1} p(s_i) N \log(N) + c_{10} \sum_{i=1}^{c_1} p(s_i) N + c_{11}$.

INSPECTCANDIDATES

Line 21 takes time less than c_{12} dbms(N) + c_{13}.

Line 22 accesses text-parsing rules. There are entity-specific rules and data instance-specific rules. The number of entity-specific rules is $c_{14} E$, where E is the number of entities. The number of instance-specific rules is $c_{15} N$, where $0 \le c_{15} \le 1$ (in fact, c_{15} is much closer to 0 than 1). If each entity has at least one data instance, then $E \le N$, and the total number of the rules $c_{14} E + c_{15} N \le (c_{14} + c_{15}) N$. The execution time for line 22 is less than dbms($(c_{14} + c_{15}) N$) = c_{16} dbms(N) + c_{17}.

Lines 23-29 take time less than a constant because they process data with a limited size.

The variable `data_instances` is a table where one dimension is limited by the number of entities, the other dimension is limited by the number of data instances. Adding an item to `data_instances` in line 32 takes time less than $c_{18} \log(E) + c_{19}$.

The variable `entities` is a table where one dimension is limited by a constant number of words in the user question and the other dimension is limited by the number of entities. In line 36, adding an entity name to `entities` takes time less than $c_{20} \log(E) + c_{21}$.

Lines 30, 31, 34, and 35 take constant time. The loop in lines 33-38 is executed up to c_4 times. Thus, lines 30-39 take time less than $c_{22} \log(E) + c_{23}$.

Lines 21-39 take the total time of lines 21, 22, 23-29, and 30-39, which is c_{24} dbms(N) + $c_{25} \log(E) + c_{26}$. If each entity has at least one data instance, then $E \le N$, and we replace $\log(E)$ with $\log(N)$, which leads to c_{24} dbms(N) + $c_{25} \log(N) + c_{26}$. Because dbms(n) is either constant or logarithmic, it dissolves in the expression. Lines 21-43 take time less than $c_{27} \log(N) + c_{28}$. The loop in lines 20-40 takes time less than $c_4 \sum_{i=1}^{c_1} p(s_i) N$ ($c_{27} \log(N) + c_{28}$). The entire routine INSPECTCANDIDATES takes time less than $c_4 \sum_{i=1}^{c_1} p(s_i) N$ ($c_{27} \log(N) + c_{28}$) + $c_{29} = c_{30} \sum_{i=1}^{c_1} p(s_i) N \log(N) + c_{31} \sum_{i=1}^{c_1} p(s_i) N + c_{29}$.

MATCHDATAINSTANCES

The execution time of the routine MATCHDATAINSTANCES is the total of the execution times of the routines EXPANDQUERY, SELECTCANDIDATES, and INSPECTCANDIDATES, which is $c_{32} \sum_{i=1}^{c_1} p(s_i) N \log(N) + c_{33} \sum_{i=1}^{c_1} p(s_i) N + c_{34}$. The parameter $p(s)$ makes such an expression difficult to read, therefore let us first estimate the value of $p(s)$.

Because $p(s)$ is a probability, its maximum value is 1, which leads to the *worst case* execution time

$$T_{\text{MATCHDATAINSTANCES}} \leq c_{35}\, N \log(N) + c_{36}\, N + c_{34}.$$

Let us assume that all stems in the data index have an equal number of attached pointers (see Figure 3.4), which roughly means that different word stems in the database are used equally often. In such a case $p(s) = c_{37}/S$, where c_{37} is the average number of words in data instances, and S is the number of unique word stems in the database. This leads to the execution time for more *favorable conditions*

$$T_{\text{MATCHDATAINSTANCES}} \leq c_{38}\, \frac{N}{S}\, \log(N) + c_{39}\, \frac{N}{S} + c_{34}.$$

The question assistant has $S > 3000$, which makes S influential for data collections with up to a few million data instances.

In the *O*-notation [Cormen at al. 1990, p. 26],

$$T_{\text{MATCHDATAINSTANCES}} = \boldsymbol{O}\,(N \log(N)).$$

4.2 Matching of Question Templates

When the matching of the user question to data instances is finished, the question assistant knows which entities and their respective instances are referred to in the question. Still, the system knows nothing about the relationships between the identified entities. That is where the help of question templates is needed: they express relationships within the ER model.

The objectives of matching the user question to the question templates – Process 2 in Figure 2.6 – are to find the templates semantically related to the question.

4.2.1 Matching Process

The matching has two parts. First, the system scans through all the question templates and approves some of them for a closer examination. An approved template references only the entities relevant to the user question, or it references no entities at all. In the latter case, the template is an ordinary FAQ. Second, the system matches the approved template to the user question. The routine MATCHTEMPLATES, which implements Process 2, embraces both parts and returns three arrays of question templates – close, related, and complementary templates (see Section 3.3.4). We should note that this routine does not apply query expansion.

MATCHTEMPLATES uses the following structure of question templates:

```
type template
  required, the array of required keywords
  optional, the array of optional keywords
  forbidden, the array of forbidden keywords
  limit, the limit of nonenvisaged words
  entities, an array of the names of the entities used in
            the template
end
```

The code of the routine follows.

```
function MATCHTEMPLATES(question,
                        templates,
                        stopwords,
                        entities)
  returns
    three arrays of question templates
  inputs
    question, the user question
    templates, the array of question templates
    stopwords, the array of stop-words
    entities, table in returns (2) of MATCHDATAINSTANCES
  locals
    token, "close", "related", "complementary", or "none"
    close, an array of question templates
    related, an array of question templates
    complementary, an array of question templates
01  for i ← 1 to ARRAYLENGTH(templates) do
      /* approve a candidate template */
02    for j ← 1 to ARRAYLENGTH(templates[i].entities) do
03      if not (templates[i].entities[j] in entities) then
04        next i
05      end
06    end
      /* match the candidate template to the user question */
07    token ← PKM(question,templates[i],stopwords,entities)
08    if token = "close" then
09      close ← close + templates[i]
10    end
11    if token = "related" then
12      related ← related + templates[i]
13    end
```

```
14    if token = "complementary" and
        ARRAYLENGTH(templates[i].entities) > 0 then
        /* the template does not match the user question */
        /* but it does refer to the same entities */
15      related ← related + templates[i]
16    end
17  end
18  return (close, related, complementary)
```

The paper "Automated FAQ Answering: Continued Experience with Shallow Language Understanding" in Part 3 of the thesis presents an informal description of the Prioritized Keyword Matching technique used to match an FAQ entry to a user question. Because the structure of an FAQ entry is similar to that of a question template, the routine PKM uses the same algorithm to match a question template to a user question, slightly modified with respect to entities. PKM returns a token which indicates how closely related the template and the question are.

```
function PKM(question, template, stopwords, entities)
  returns
    "close", "related", "complementary", or "none"
  inputs
    question, the user question
    template, a question template
    stopwords, the array of stop-words
    entities, the table from the input of MATCHTEMPLATES
  locals
    match_ok, "true" or "false"
    words, an array of words
    m, the number of unique words in question
    n, the number of unique words in question that match
        the keywords in template

    /* process forbidden keywords */
19  for i ← 1 to ARRAYLENGTH(template.forbidden) do
20    match_ok ← MATCHKEYWORD(question,template.forbidden[i])
21    if match_ok then return "none"
22  end
23  clear the marks in question to be made again in
    lines 32 and 59
```

```
    /* process required keywords */
24  for i ← 1 to ArrayLength(template.required) do
25    (match_ok, question)
        ← MatchKeyword(question, template.required[i])
26    if not match_ok then return "complementary"
27  end
    /* process optional keywords; no match_ok this time */
28  question ← MatchKeyword(question, template.optional)
    /* process stop-words */
29  words ← split question into words
30  for i ← 1 to ArrayLength(words) do
31    if words[i] in stopwords then
32      mark the i^th word in question
33    end
34  end
    /* process the limit of nonenvisaged words */
35  m ← count the total number of unique words in question
36  n ← 0
37  words ← split question into unique words
38  for i ← 1 to ArrayLength(words) do
39    if words[i] is marked in lines 32 or 59 then
40      n ← n + 1
41    else
42      for j ← 1 to ArrayLength(template.entities) do
43        if template.entities[j] in entities[words[i]]
          then
          /* words[i] matched a data instance relevant for this template */
44          n ← n + 1
45        end
46      end
47    end
48  end
49  if m – n > template.limit then
50    return "related"
51  else
52    return "close"
53  end
```

```
function MatchKeyword(question, keyword)
  returns
    1) "true" or "false"
    2) question with marked words
  inputs
    question, the user question
    keyword, a keyword as a number of synonyms
  locals
    synonyms, an array of the keyword's synonyms
    match_ok, "true" or "false"
54  match_ok = "false"
55  synonyms ← split keyword into synonyms
56  for i ← 1 to ArrayLength(synonyms) do
57    if synonyms[i] matches question then
58      match_ok = "true"
59      mark the matching words in question
60    end
61  end
62  return (match_ok, question)
```

4.2.2 Estimated Execution Time

The time needed to select relevant question templates depends on the size of user questions, the size of question templates, and, most of all, the number of question templates. In order to find out which entities are relevant for a given template, the routine searches through the array of the names of the entities referenced by the user question. All user questions are one sentence long and contain a limited number of words. Each question template stands for a one sentence long question and has a limited number of keywords, their synonyms, and referenced entities. If the total number of question templates is Q, and the total number of entities is E, the execution time is a function $T(Q, E)$.

In the routine PKM, the only data item with formally unlimited size is ***entities*** processed in line 43. Line 43 takes time less than $c_1 \log(E) + c_2$. The rest of PKM, as well as MatchKeyword, take time limited by a constant because the size of the data they process is limited by a constant. The entire routine PKM takes time less than $c_1 \log(E) + c_3$.

In the routine MatchTemplates, line 3 takes time less than $c_4 \log(E) + c_5$. Because the number of entities referenced in a template is limited, the loop in lines 2-6 takes time less than $c_6 \log(E) + c_7$. Lines 8-16 take constant time. Lines 2-16 take time less than $c_6 \log(E) + T_{PKM} + c_8 = c_9 \log(E) + c_{10}$. The loop in lines 1-17 is executed Q times and takes time less than $Q\,(c_9 \log(E) + c_{10})$. The entire routine takes time

$$T_{\text{MATCHTEMPLATES}} \le c_9\, Q \log(E) + c_{10}\, Q + c_{11}.$$

The question assistant currently operates 6 entities and about 40 question templates. Assuming that we can always ask more than one question about an entity, we expect that $E \le Q$. Then

$$T_{\text{MATCHTEMPLATES}} \le c_9\, Q \log(Q) + c_{10}\, Q + c_{11}.$$

In the *O*-notation,

$$T_{\text{MATCHTEMPLATES}} = \boldsymbol{O}\,(Q \log(Q)).$$

4.3 Creating Interpretations of the User Question

Process 1 in Figure 2.6 returns an array of data instances. Process 2 returns an array of question templates. Separately the instances and templates are not meaningful for the user. Therefore Process 3 joins them: the system replaces the entity slots in the question templates with the data instances and obtains a number of questions – the system's own interpretations of the submitted user question. Because each entity slot represents an entity, and each data instance belongs to an entity, this task poses no algorithmic difficulties.

Likewise, the system replaces the entity slots in the associated database query templates and obtains executable queries which create the answers of the interpretations.

4.3.1 Credibility Tests

Unfortunately, not all interpretations may prove meaningful. For example, "When does Pavarotti sing in Globen?" could match two locations with the same name: the arena "Globen" and the gift shop "Globen". We know that an arena is more appropriate for concerts than a shop. The system does not know that until it validates the interpretations.

For each interpretation, the system runs a *credibility test*: it executes the database query associated with the interpretation. If the query returns a positive result (e.g., the artist has at least one concert scheduled in that place), the interpretation is meaningful. Otherwise the interpretation is either meaningless or useless because it has no positive answer anyway.

We must keep in mind, however, that if the question assistant rejects all meaningful interpretations because they have no answer, the system may leave a wrong impression that it has not "understood" the user question. The system must find the way to deal with it. For example, to skip credibility tests for some templates under certain conditions.

The routine `TestInterpretations`, which implements a part of Process 3, performs credibility tests.

```
function TestInterpretations(question_templates,
                             query_templates,
                             data_instances)
  returns
    an array of interpretation-query pairs
  inputs
    question_templates, an array of question templates to
        be tested
    query_templates, the array of database query templates
    data_instances, data instances from MatchDataInstances,
        see returns (1) in MatchDataInstances
  locals
    query_template, a database query template
    combinations, an array of combinations of data
        instances
    query, a database query
    dummy, some result of the execution of query
    interpretation, a natural language question
    pair, coupled interpretation and query
    pairs, an array of pair
01  for i ← 1 to ArrayLength(question_templates) do
02    query_template ← select a database query template
                       from query_templates associated with
                       question_templates[i]
03    combinations ← Cartesian product of data instances
                     passed for testing in data_instances
                     /* returns (1) in MatchDataInstances show */
                     /* the table structure of data_instances */
04    for j ← 1 to ArrayLength(combinations) do
05      query ← replace the entity slots in query_template
                with data instances in combinations[j]
06      dummy ← execute query /* test for the first item only */
07      if dummy is not empty then
08        interpretation ← replace the entity slots in
              question_templates[i] with the data instances
              from combinations[j]
09        pair ← interpretation + query
10        pairs ← pairs + pair
11      end
12    end
13  end
14  return pairs
```

4.3.2 Estimated Execution Time of Credibility Tests

The time needed for credibility tests depends on the number of question templates to be tested, q, the number of the data instances that replace the entity slots, n, and the complexity of the involved database query templates, t. The execution time is a function T(q, n, t).

Line 2 takes constant time if question templates and database query templates are linked together. In line 3, the number of items in `combinations` is $\prod_{i=1}^{m} n_i$, where m is the number of entities referenced in the given template, and n_i is the number of data instances to be tested for the i^{th} entity slot. The number m is limited by a constant c_1 (we expect $c_1 \le 2$). Line 3 takes time less than $c_2 (\prod_{i=1}^{c_1} n_i) + c_3$. Both lines 2-3 take time less than $c_2 (\prod_{i=1}^{c_1} n_i) + c_4$.

The time needed to execute a database query depends on the DBMS and the complexity of the query. We have no information about the complexity of an arbitrary query, its execution time is designated with t. Line 6 takes time t, lines 5 and 7-11 take constant time, therefore lines 5-11 take time $t + c_5$. The loop in lines 4-12 takes time less than $(\prod_{i=1}^{c_1} n_i)(t + c_5)$.

Lines 2-12 take the total time of lines 2-3 and 4-12, which is less than $(\prod_{i=1}^{c_1} n_i)(t + c_6) + c_4$. If q is the number of question templates to be tested, then the loop in lines 1-13 takes time less than $(\sum_{j=1}^{q}(\prod_{i=1}^{c_1} n_{ij})(t_j + c_6)) + c_4\, q$, where n_{ij} is the number of data instances to be tested for the i^{th} entity slot of the j^{th} question template. The entire routine takes time

$$\mathrm{T}_{\texttt{TestInterpretations}} \le (\sum_{j=1}^{q}(\prod_{i=1}^{c_1} n_{ij})(t_j + c_6)) + c_4\, q + c_7.$$

Let us limit t_j with a constant. The maximum n_{ij} is N – the total number of data instances in the database. The maximum q is Q – the total number of question templates. We expect that the database have more data instances than people have questions about them, therefore $Q \le N$. This gives us the worst case execution time

$$\mathrm{T}_{\texttt{TestInterpretations}} \le c_8 N^{c_9} + c_4 N + c_7.$$

In the O-notation,

$$\mathrm{T}_{\texttt{TestInterpretations}} = \boldsymbol{O}(N^c),$$

where c is some constant.

The worst case execution time makes no sense because it suggests that Processes 1 and 2 in Figure 2.6 return the entire target database. In average, n_{ij} and q lie between 1 and 10, and c_1 is 1 or 2.

4.3.3 How to Reduce the Time of Credibility Tests

We can reduce $T_{\text{TESTINTERPRETATIONS}}$ by reducing q, n_{ij}, and t_j.

Reducing q. Certain question templates, especially those referencing only one entity, do not require credibility tests. For example, "Where is <location-name>?" is meaningful with any location name. The situation is different for the templates that reference several entities. There may be semantically conflicting combinations of data instances in the referenced entities. Credibility tests detect such conflicts.

Reducing n_{ij}. Improving the matching techniques for data instances, first of all entity-specific text-parsing rules, reduces the number of irrelevant data instances passed for the test. We can reduce the number of the data instances if we define more specific entities and more specific question templates.

Reducing t_j. The execution time of an arbitrary query heavily depends on the DBMS being used. More sophisticated knowledge domains and database queries require a more efficient DBMS. Also, it helps if the DBMS has a facility which checks whether or not the query returns data without actually retrieving the data.

4.4 Summary

After the user has selected and resubmitted an interpretation of the original question (Process 4 in Figure 2.6), the system executes the associated database query (Process 5). The query returns raw data which is formatted and complemented with wrapping text, such as a header and footer, specific for each database query template. Processes 4 and 5 pose no algorithmic difficulties.

More interesting are Processes 1 and 2. Their worst case execution time is $T_{\text{MATCHDATAINSTANCES}} + T_{\text{MATCHTEMPLATES}} = \boldsymbol{O}(N \log(N)) + \boldsymbol{O}(Q \log(Q))$, where N is the number of data instances in the target database, and Q is the number of question templates. In Section 4.3.2 we stated that $Q \leq N$. Hence, estimated execution time for Processes 1 and 2 is

$$T_{1+2} = \boldsymbol{O}(N \log(N)).$$

We may observe that this execution time is equivalent to that of popular sorting algorithms [Cormen at. al 1990, p. 138].

Credibility tests are an optional part of Process 3. Their execution time is dominated by $\sum_{j=1}^{q}(\prod_{i=1}^{c_1} n_{ij})\, t_j$, where q is the number of question templates to be tested, n_{ij} is the number of the data instances to be tested for the i^{th} entity slot of the j^{th} question template, and t_j is the execution time of the database query of the j^{th} template. In the unreal worst case, estimated execution time for credibility tests is

$$T_3 = \boldsymbol{O}(N^C).$$

5 Maintenance of the Question Assistant

The complexity of NLP has limited the scope of automated question answering to a number of domain-specific systems. Only recently have there appeared question-answering systems, such as Ask Jeeves[34] and Mulder [Kwok at al. 2001], which scale question answering to the entire WWW. Ask Jeeves is the first large-scale commercially successful question-answering system, its technology resembles that of this research (see Section 7.4). Mulder is a research prototype which automatically processes and extracts answers form Web pages.

The solutions of AI are not acceptable in many organizations because they require high qualifications and rare skills of the involved personnel. The question assistant pursues a different question-answering approach. Why should it be successful? The main issue that determines maintainability of a question-answering system is the amount and competence of the human resources needed in order to set up and run a new instance of the system. The question assistant does not require rare skills, as it will be discussed in this section.

The question assistant is still a research prototype which answers questions on events in Stockholm, the capital of Sweden. The entity definitions comprise 6 entities with about 11 000 data instances (i.e., the names of objects in 6 entities are indexed and recognized; there are more than 6 attributes used in answers of user questions). There are about 40 question templates. 226 user questions were considered for evaluation of the performance of the system (see Section 6).

5.1 Setting Up and Maintaining the Question Assistant

The report “Application and Maintenance Aspects of an FAQ Answering System”, included in Part 3 of this thesis, discusses the maintenance of two FAQ answering systems. The question assistant – a descendant of those two systems – has similar maintenance procedures.

5.1.1 Selecting Entities

The entities involved in question answering are selected manually by reading the ER model. Test the appearance of the data instances in natural language sentences! Do they sound good? Having ISA and PartOf relationships, we may choose to use a parent of a number of synonym entities (see Section 3.3.6). On the other hand, we may need to split an entity because its data instances do not sound equally good in the same natural language sentence. For example, the question assistant has two entities for performers: performers-artists (singers, dancers, etc.) and performers-non-artists (lecturers, talk show members, etc.).

[34] http://www.askjeeves.com/, valid in November 2001

After the entities are chosen, they are defined for the question assistant using the editing form for entity definitions (see Section 3.1). In an entity definition, an SQL statement selects the data instances that belong to this entity.

Entity-specific text-parsing rules for the data instances, which usually are names of real world objects, are highly desired (see Section 3.1.3). Built-in rules are preferred. It is easy to select a built-in rule if it is already implemented (see Section 3.1.4 about reusability of the rules). Implementation of a different built-in rule requires programming and cannot be done by an ordinary administrator of the question assistant. If the structure of the text is very regular, a custom rule may be sufficient. Custom entity-specific text-parsing rules are easy to create and modify because they are composed as combinations of sequence numbers of words.

If no text-parsing rule is specified for the entity, the question assistant applies the "ruleless" algorithm in order to match the data instance to the user question. Currently the quality of such matching is mediocre, yet improvements are possible.

X-rules are data instance specific and override any other text-parsing rules. Ideally, the administrator scans through the data instances and finds out where x-rules are needed. For example, an artist may have a pseudonym which is introduced in an appropriate x-rule (example in Figure 3.3). It may turn out a tedious work to check a large number of data instances in order to introduce x-rules. To our knowledge, there is no other way to treat exception as to detect them and define how to handle them. If we want to skip scanning the data instances, we may wait until a user asks a question that proves that a particular entity-specific rule does not work properly, a data instance did not match properly, and an x-rule is needed.

X-rules are rather helpful for entities with a small number of data instances. The question assistant has an entity for 41 category of events registered in the SIS database (e.g., "exhibitions", "auctions & markets", "children"). Each of the categories has its own x-rule, which allows effective discovering of areas of interests implied in user questions.

5.1.2 Creating Question Templates

Question templates represent relationships, expressed in natural language, between entities and their attributes, as well as between entities themselves. A question template may represent one or two, not likely more, binary relationships. A question template may consider one or several attributes of an entity in the answer. For example, "What is the admission fee for `@event`?" encompasses one attribute – price, whereas "How can I contact `@place`?" encompasses several attributes, such as telephone number, street address, e-mail address, and URL.

Unlike FAQs, question templates are likely to reference time. If an entity has a time attribute, most probably this attribute must be included in the template. Even if people do not mention time explicitly, they usually imply some: this week, this

weekend, nearest future. The system must maintain a default time period in case if time is implied but not explicitly mentioned in the user question.

The process of creating a question template following:

1. Select the entities or the entity and its attributes that are the objects of the question.
2. Select the relationships and formulate the question. Think whether the existing selection of entities is appropriate for different relationships between the given entities: Are there synonym entities that can be combined? Do we need to split this entity in two to make the data instances sound good in the question?
3. Consider the time attribute and include appropriate entity slots if necessary.
4. Prepare the question template for Prioritized Keyword Matching as described in Section 3.2 in Part 2 of this thesis.

5.1.3 Creating Answer Templates

Before an answer template is created, we have to figure out what information is needed in order to answer the question and which physical data structures – SQL tables – carry that information. We must take into account requests for both explicit and implied information. For example, the answer to "Has U2 any concert in Stockholm?" is expected to contain not only "yes" but also the time and location of the concerts.

The database query template (see Section 3.4.1), which is a stored procedure on a Microsoft SQL Server, is the main part of the answer template. The SQL query of the stored procedure consists of a *select list* – "SELECT $column_1$, $column_2$, … $column_n$ FROM $table_1$, $table_2$, … $table_m$" – and a *where-clause*. After the SQL query is executed, "$column_1$, $column_2$, … $column_n$" contain the data instances that form the answer. The where-clause is the search condition that determines which data instances are to be selected. As a rule, the where-clause makes use of entity slots – the input parameters of the stored procedure.

In the answer layout template (see Section 3.4.2), the main decision is how to format the data instances retrieved by the database query template:

- If we expect no more than one data instance in each column of the select list to be printed in a comma-separated list (Figure 3.9, only one item in the comma-separated list), we use the `<answer_src>` tag with no parameters.
- If we expect no more than one data instance in each column of the select list to be printed together with explanatory text (Figure 3.11), we combine the `<answer_src>` and `<answer_clause>` tags.
- If we expect more than one data instance in some column of the select list (Figure 3.10), a table is the most appropriate format for printing the data

instances. In such a case we use the <answer_src | heading$_1$ | heading$_2$ | ... | heading$_n$> tag.

5.1.4 Maintenance Routines

Defining entities and creating question templates is an iterative process: we define entities, involve them in question templates, then refine the entities to adapt them to the question templates, then create next question templates, and so on.

After a new instance of the question assistant is established, the initial collection of the question and answer templates has to be created before the system is put into operation. The initial question templates mirror the main relationships within the ER model of the information system. It is better to ask tolerant users to run the first testing, because at this stage many questions are likely to remain unanswered. After fine-tuning is done, the system can be announced to a broader audience.

When the question assistant is in operation, the main duty of the administrator is to monitor transaction logs which contain the user questions and the system's replies. Maintenance of the question assistant implies improving the question templates and improving the quality of the retrieval of relevant data instances residing in the target database. Possible causes of retrieving irrelevant data instances can be, among others:

- A particular stem in the repository of stems or the repository of irregular forms is too indistinct and, therefore, introduces wrong synonymy. *Cure:* Refine the stem into more specific stems.
- Two or more data instances that contain common words are repeatedly retrieved together where it is not appropriate. Most probably, this is a failure of the "ruleless" algorithm of matching data instances to a user question. *Cure:* Introduce x-rules which define how to match the text of each specific data instance to a user question.
- The user question contains proper names in Swedish, but the question assistant retrieves inappropriate data instances in English. Possibly, the wrong data instances in English are translations of the counterparts in Swedish. This means that the system has matched wrong data instances in Swedish. *Cure:* Check the above two causes.

Given that the retrieved data instances are correct, user questions may be poorly answered because of imperfect or missing question templates:

- If an on-topic user question has no corresponding question template, create such a template. Consider two options:
  - if there exists an answer template that fits the question template to be created, then add an auxiliary template to the existing cluster of question templates that has the right answer template;

  - otherwise, create a new cluster of question templates with a new answer template.

- If a wrong question template is retrieved, or it is incorrectly classified as close or related interpretation, then:
  - adjust the required, optional, and forbidden keywords, as well as the limit of nonenvisaged words so that the template does not match inappropriate user questions or is correctly classified;
  - consider refinement of this question template into two or more auxiliary templates which may belong to different clusters (i.e., the similar in appearance questions may have different answers).

5.2 Manual Maintenance Procedures

The question assistant is completely autonomous when it answers user questions. Yet it does require human input in order to transform the knowledge from an ER model to entity definitions and question templates. Section 3.4 in Part 5 of this thesis answers the question why question templates are created manually. Intelligent decision making during the process of creating of question templates takes human reasoning because today's technologies cannot communicate the meaning of a natural language sentence from the human mind to the computer without any manual work done at some point of the development of the system itself or its imported components. The manual knowledge transformation requires that the administrator of the system understand conceptual modeling and database design, is familiar with the subject area of the system, and has a good command of the English language.

Skills in conceptual modeling and database design, as well as understanding of the subject area are necessary in order to read the ER model and:

- extract the entities to be used by the question assistant and define their scope in the implemented database by means of formal database (e.g., SQL) queries;
- extract the relationships between the entities, as well as between the entities and their attributes, in order to build question templates;
- create efficient database query templates – parameterized database queries – which retrieve data instances used in the answer of a user question.

Command of English is necessary in order to prepare question templates for Prioritized Keyword Matching. The administrator:

- considers alternative forms of a question and selects a number of auxiliary templates in a cluster of question templates;

- selects representative required, optional, and forbidden keywords for each question template considering various connotations of the question and the background of expected users;
- selects context dependent synonyms considering their inflections for each keyword;
- adds missing entries to the repositories of stems and irregular forms (fortunately, fewer and fewer new stems and irregular forms are needed as the repositories get filled up).

This is how the system “learns” the questions relevant for the target database and their meanings.

There exist tools that assist in the process of creating and maintaining of question templates. See the chapter “Manual Routines vs. Automated Routines, Tool Support” in the paper “Application and Maintenance Aspects of an FAQ Answering System” in Part 3 of this thesis for information about tool support.

5.3 Summary

Maintenance of the question assistant has two focuses: maintenance of entity definitions and maintenance of question / answer templates.

When the administrator of the system creates entity definitions, he or she reads the ER model and selects relevant entities. After the entities are selected, they are defined for the question assistant. In an entity definition, an SQL statement selects the data instances that belong to this entity. An entity-specific text-parsing rule tells the system how to match the data instances to a user question. X-rules are data instance specific and applied when the entity-specific rule does not fit.

Question templates represent relationships, expressed in natural language, between entities and their attributes, as well as between entities themselves. When the administrator creates a question template, he or she selects one or several entities, their attributes if necessary, the relevant relationships, and writes down the question. Entity slots correspond to the selected entities. After this, the administrator prepares the question template for Prioritized Keyword Matching.

When the administrator creates an answer template, he or she finds out which data is needed in order to answer the question and which physical data structures carry that data. After this, he or she creates the database query template – a formal database query with entity slots – and the answer layout template.

When the question assistant is in operation, the main duty of the administrator is to monitor transaction logs which contain the user questions and the system’s replies. Maintenance of the question assistant implies improving the question templates and improving the quality of the retrieval of relevant data instances residing in the target database.

Maintenance of the question assistant does not require rare skills: the system is to be maintained by a database developer who is familiar with the subject area of the system and has a good command of English.

6 Performance of the Question Assistant

Performance of the question assistant was evaluated by measuring its ability to retrieve all the relevant information and only relevant information. The system uses two independent techniques to retrieve data instances from the target database and to retrieve question templates. Therefore the performance of both techniques was measured separately.

226 different user questions in English were considered. The questions in Swedish were disregarded.

The decisions on which retrieved information was relevant and which was irrelevant were made by the author of this thesis alone and, therefore, were subjective.

6.1 Recall, Precision, and Rejection

In Information Retrieval, there are two parameters to measure the quality of document retrieval. *Recall* is the system's ability to retrieve all the relevant documents:

$$Recall = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ relevant\ documents\ in\ the\ collection} .$$

Precision is the system's ability to retrieve only relevant documents:

$$Precision = \frac{Number\ of\ relevant\ documents\ retrieved}{Total\ number\ of\ documents\ retrieved} .$$

Recall is defined only for the user queries that have corresponding documents in the collection. It is not clear, however, how we should perceive precision if the corresponding recall is 0. For an average Information Retrieval system this is not a problem: it usually retrieves some relevant documents. Also, it is not clear how we should perceive precision for the user queries whose recall is not defined: it makes no sense to calculate the proportion of relevant documents if we know there are no relevant documents. In this evaluation, precision was measured only for the user queries whose recall was defined and not 0.

Another interesting quality is the system's ability not to retrieve garbage if there are no relevant documents in the collection. [Burke et al. 1997-a] proposes the measure called *rejection* – the proportion of user queries that the system correctly reports as having no corresponding documents in the collection:

$$Rejection = \frac{\begin{array}{c}Number\ of\ user\ queries\ that\ have\ no\ relevant\ documents\\ in\ the\ collection,\ and\ no\ documents\ are\ retrieved\end{array}}{\begin{array}{c}Total\ number\ of\ user\ queries\ that\ have\ no\ relevant\\ documents\ in\ the\ collection\end{array}}.$$

Consequently, the rejection value 1 means that the system does not retrieve irrelevant documents if user queries have no relevant documents in the collection (i.e., the system properly gives the "not found" message). The rejection value 0 means that the system always retrieves irrelevant documents if user queries have no relevant documents in the collection. Rejection is analogous precision, yet its formal definition is different. Also, good precision may be accompanied with bad rejection, as we will see it further.

6.2 Retrieval of Data Instances

Measurements

Recall. 135 user question were considered for the measurements of the recall of the retrieval of data instances residing in the target database. The other 91 user questions did not reference any data instances in the target database and, therefore, were disregarded. The average recall was 0.82.

Precision. 117 user questions were considered for the measurements of the precision of the retrieval of data instances residing in the target database (18 cases had recall 0; 135 -18 = 117 questions considered). The average precision was 0.83.

Rejection. 91 user questions that did not reference any data instances were considered for the measurement of rejection with respect to data instances residing in the target database. In 52 cases no data instances were retrieved, and in 39 cases irrelevant data instances were retrieved. The rejection was 52 / 91 = 0.57.

What Influences the Performance?

The entities with the worst recall were those dealing with the names of events ("event") and places ("place"). The entity with the worst precision was "place". In both cases a "ruleless" *matching algorithm*, one for both entities, was used (see Section 3.1.3).

Whatever matching algorithm is used, precision and rejection are influenced not only by the capabilities of the matching algorithm but also by *retrieval of the right question templates* and *credibility tests*. The matching algorithms may select a large number of irrelevant data instances that pertain to irrelevant question templates. If the system does not retrieve those irrelevant question templates, the garbage is never presented to the user. Likewise, if an irrelevant data instance fails the credibility test, it is never presented to the user. Unfortunately, credibility tests became rather slow after the SQL "join" statements appeared in database query templates. Rather few data instanced manage to undergo credibility tests before the

time-out (6 seconds) occurs, which negatively influences the precision of the retrieval of data instances.

We may observe that precision (0.83) is higher than rejection (0.57), which means that the system is less inclined to retrieve irrelevant data instances if user questions do reference data instances residing in the target database. This is influenced by bad rejection with respect to question templates (i.e., the system often retrieves irrelevant question templates if there are no relevant templates) and good precision of the retrieval of question templates (i.e., the system retrieves mostly relevant question templates if there are some).

Confusing *categorization of data instances* in the target database negatively influences both recall and precision. For example, in the SIS database "Water Festival" is classified as a place rather than an event. Consequently, it is not presented to the user who asks "Will there be the Water Festival?" because the user question does not match any question templates dealing with places. Further, "Janet Jackson" and many other artist names are classified as names of events (i.e., the concerts of the artists are meant) rather than names of people, therefore the matching algorithms designed for entities dealing with person names are not applied.

Bad *spelling checking* leads to bad precision. If the question assistant does not recognize a word, the system attempts to alter its spelling. Because spelling correction is not in the focus of this research, the system has no good spelling correction algorithm. The means of the Microsoft SQL server are used instead. Those means provide a good input for a more rigorous spelling correction algorithm, but they should not be used instead of such an algorithm.

6.3 Retrieval of Question Templates

Measurements

Recall. 143 user question were considered for the measurements of the recall of the retrieval of question templates. The other 83 user questions had no corresponding question templates and, therefore, were disregarded. The average recall was 0.98.

Precision. 142 user questions were considered for the measurements of the precision of the retrieval of question templates (in one case recall was 0). The average precision was 0.93.

Rejection. 83 user questions that had no corresponding question templates were considered for the measurement of rejection with respect to question templates. In 40 cases no question templates were retrieved, and in 43 cases irrelevant question templates were retrieved. The rejection was 40 / 83 = 0.48.

What Influences the Performance?

Mainly Prioritized Keyword Matching is responsible for the quality of the retrieval of question templates (see the chapter “What is a Good FAQ Entry?” in the paper “Automated FAQ Answering: Continued Experience with Shallow Language Understanding” in Part 3 of this thesis).

Besides, precision and rejection with respect to question templates improves if we improve precision and rejection with respect to data instances residing in the target database. If the system does not retrieve irrelevant data instances, then fewer irrelevant entities and question templates are considered. Consequently, fewer mistakes are made.

6.4 Summary

The measurements of the performance of the question assistant are summarized in Table 6.1.

Table 6.1 Recall, precision, and rejection

Retrieval of	Recall	Precision	Rejection
data instances	0.82	0.83	0.57
question templates	0.98	0.93	0.48

The recall and precision in the above table can be expressed in terms of probabilities. The probability that the required data instance is retrieved is 82%, the probability that the retrieved data instance is one we are looking for is 83%. These are reasonable figures, although better recall is desired. The probability that the required question template is retrieved is 98%, and the probability that the retrieved question template is one we are looking for is 93%. These are good figures.

The figures of rejection show that the question assistant performs not that well when users ask questions with no answers, i.e., when the system is not capable to answer the question due to lack of relevant question templates or data instances in the target database.

In Information Retrieval, usually there is a trade-off between recall and precision: higher recall yields lower precision, and vice versa. [Salton and McGill 1983, p.170] presents a series of average recall-precision measurements for an Information Retrieval system. Figure 6.1 shows the graph (long thin line) that represents the recall-precision dependencies for that system.

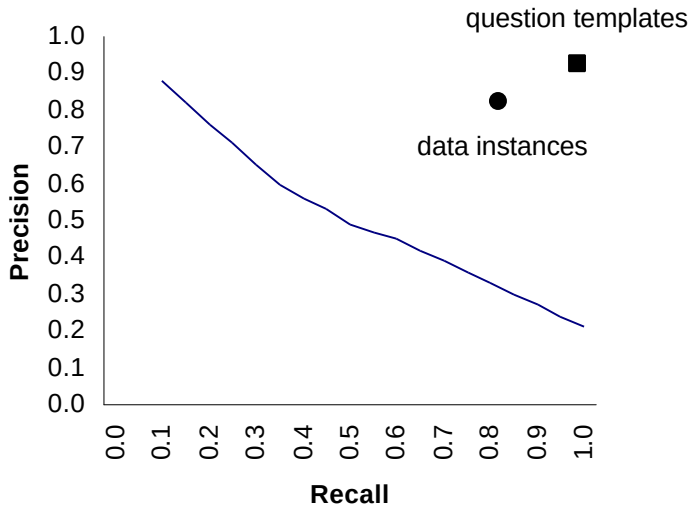

Figure 6.1 Recall-precision graphs.

This research did not investigate the recall-precision dependencies for the question assistant because there are too many parameters that influence these dependencies. The average recall and average precision were measured once for data instances and once for question templates. In Figure 6.1, the only recall-precision links are represented by a small circle (for data instances) and a small square (for question templates).

The question assistant is still a research prototype, and there has been only one public test case. The test results have shown the strengths and weaknesses of the current implementation of the system, they have drawn the directions of short-term development. The very first test case cannot be used as a proof of the capabilities of the ideas lying in the foundation of the question-answering approach.

7 Related Techniques and Systems

This section compares the question assistant with related techniques and systems: (i) Information Retrieval which works with large free-text documents, (ii) Natural Language Processing which performs semantic parsing of user queries and uses ontologies, (iii) interfaces based on the information system's conceptual model, and (iv) Ask Jeeves.

7.1 Information Retrieval

The question assistant is not an Information Retrieval system [Salton and McGill 1983; Salton 1989, pp. 227-424] first of all because the question assistant transforms a user question into a formal database query whereas Information Retrieval systems never do that. The question assistant deals with data instances in the target database, which is a structured database. Information Retrieval deals with free text documents only, never with data instances in a structured database. The lengths of user queries are different: the question assistant is designed to answer 1-10 words long natural language sentences, whereas Information Retrieval deals with free-text documents that contain tens or, better, hundreds of words.

Still, the question assistant and Information Retrieval systems share some common features:

- Both the Information Retrieval systems and the question assistant have collections of free text documents / questions. As a free text user query is submitted, the systems retrieve the documents / questions relevant to the query.
- Retrieval of the free-text documents and the question templates is based on keyword comparison. Neither Information Retrieval systems nor the question assistant perform semantic analysis of user queries.
- The question assistant uses truncated words, which often happen to be word stems, in order to match the words ignoring inflections. Modern Information Retrieval systems do the same.
- Both the question assistant and Information Retrieval systems have a stop-list for the words that are too common in the given application-specific knowledge domain.
- The index of data instances used by the question assistant is similar to inverted index or inverted file in Information Retrieval.

The question assistant uses Prioritized Keyword Matching to match a user question to a question template. Table 7.1 shows the main differences between Information Retrieval and Prioritized Keyword Matching.

Table 7.1 Information Retrieval vs. Prioritized Keyword Matching

Information Retrieval	Prioritized Keyword Matching
Length of the query and documents	
The techniques of Information Retrieval were designed to work with documents and user queries longer than a few words.	The question assistant was designed to work with one-sentence-long documents and user queries.
How is the importance of a term denoted?	
The importance of a term in a term-vector is denoted primarily by its frequency in the document. It is possible to adjust the significance of the frequency by a relative weight of the term. Nonetheless, the context of a relative weight is rather broad – the entire collection of the documents. Further, relative weight is an inaccurate measure unless weighing is done manually.	The importance of a term is denoted by a manually assigned role – required, optional, or forbidden keyword – in the context of the particular question template.
Word classes	
There are open and closed word classes. Open word classes – nouns, verbs, adjectives, and adverbs – expand by new words as the language evolves. Closed word classes - prepositions (e.g., “on”, “by”), determiners (e.g., “a”, “the”, “some”), pronouns (e.g., “he”, “she”), conjunctions (e.g., “and”, “or”), auxiliary verbs (e.g., “is”, “can”, “may”), particles (e.g., “up”, “down”), numerals (“two”, “three”) – are unlikely to acquire new words.	Open and closed word classes are *not* analogous to required, optional, and forbidden keywords. Nonetheless, we can observe that the question assistant’s list of stop-words contains closed class words and those open class words that are very common within the subject of the question assistant.
What gets indexed?	
Both the user query and the documents in the collection are indexed.	Only question templates are indexed.
Common thresholds vs. individual properties of question templates	
There is a common threshold or cut-off line used in order to decide whether or not a document in the collection is relevant to the user query. One threshold is empirically determined for the entire collection of the documents.	The result of matching a question template to a user question is based on individual properties of each question template.

Scope, context of synonymy	
The system can bring a number of synonyms to a common word or a word stem. The context of the synonymy is the entire collection of documents.	Synonymy of the keywords of a particular question template is resolved in the context of this question template. Synonymy of the words in data instances in the target database is resolved in the context of the entire target database.

7.2 Natural Language Processing

Unlike NLP systems [Russell and Norvig 1995, pp. 691-723], the question assistant performs no semantic language processing during answering a user question. The question assistant matches the user question to (i) data instances residing in the target database, which may imply limited analysis of the words, and (ii) a number of question templates, which is done by Prioritized Keyword Matching. The question assistant does, however, require semantic analysis of the questions in the question templates at the moment of creating and updating these templates.

Table 7.2 shows the main differences between NLP systems and the question assistant.

Table 7.2 Natural Language Processing vs. Question Assistant

NLP Systems	Question Assistant
Focus of the manual work	
NLP is a branch of AI. AI focuses on teaching a question-answering system to learn and answer questions automatically, therefore it takes a large amount of manual work (creating of domain-specific lexicons and inference rules, tuning of machine learning algorithms) completed before the system is put into operation. Errors made during this stage affect the entire functioning of the system.	The question assistant is manually taught only the questions it must recognize, and their answers. The knowledge of the question assistant is augmented gradually (i.e., new question templates are created and old ones updated) before and *during* the operation of the system. Small pieces of manual work are evenly spread throughout the system's lifecycle. The question assistant is ready to work as soon as the software is installed and the first question template in the database is created. Because question templates are independent, errors made in one template do not affect other templates.

Type of the lexicon	
Word sense ambiguity is one of the biggest problems of NLP lexicons operated in the application-specific knowledge domain. An AI question-answering system must have a complicated knowledge base in order to understand the context of the word and correctly decide on its synonyms in each particular case.	The question assistant uses a multiple lexicon (see the chapter "Idea of Multiple Lexicon" in the paper "Automated FAQ Answering: Continued Experience with Shallow Language Understanding" in Part 3 of this thesis) for matching the user question to a question template. The multiple lexicon resolves synonymy in the context of the question in each question template. There still remain problems in processing the text of data instances residing in the target database.
Machine learning	
Some AI question-answering systems, such as [Winiwarter 1999], use machine learning of new questions. A positive aspect of machine learning is that it accumulates small efforts of many users. A negative aspect is that the input of end users is not reliable. The people who do surveys know that a respondent expects some kind of reward for his or her contribution (e.g., a movie ticket); there is not much possibility for a machine learning system to distribute rewards.	The question assistant uses no machine learning, and there are no such plans for the nearest future. The administrator of the system learns the subject and updates the question templates accordingly. No explicit feedback from end users is expected (still, the user questions are logged and analyzed).
Built-in vs. add-on automatization	
AI question-answering systems have built-in automatization of learning and answering questions.	The question assistant has add-on automatization of administration and maintenance of question templates (see the chapter "Manual Routines vs. Automated Routines, Tool Support" in the report "Application and Maintenance Aspects of an FAQ Answering System" in Part 3 of this thesis).
Skills of the personnel	
AI solutions require high qualifications and rare skills of the involved personnel.	The question assistant is to be maintained by a database developer who is familiar with the subject area of the system and has a good command of English.

The above comparison of NLP systems and the question assistant does not imply that any of them is absolutely superior. We may want to upgrade the techniques currently employed by the question assistant with other, more powerful techniques. We must keep in mind, however, that there will always be a trade-off between the power of natural language processing and the complexity of the system.

7.3 Graphical Interfaces Based on the Conceptual Model of the Underlying Information System

The graphical query language *QBD* (Query by Diagram) [Angelaccio et al. 1990] is a visual navigation tool on ER diagrams. The basic idea is to decompose a query into its elementary steps, expressed by a limited set of graphical operations, such as choice of icons, selection of concepts, and navigation. In order to express formally the syntax and the semantics of the language, a one-to-one correspondence between the graphical operations and the syntactic constructs of a textual query language has been defined.

By the means of the graphical interface, a user builds a query on the ER diagrams of the underlying relational database. Then the ER query is translated into a relational expression and further into a relational query in terms of the target database management system.

The *Graphical Data Manipulation Language for en Extended Entity-Relational Model* [Czejdo et al. 1990] proposes similar solutions.

V is a two dimensional, visual, direct manipulation query language designed for a multimodal environment that includes also a natural language processing component [Bretan 1995, pp. 20-29]. As natural language interfaces often do not provide enough explicitly organized structure that users can navigate through, the added visual means fix the fault. The syntax of V is based on that of the ER model. The main emphasis of the language regarding visuality is on visualization of the logical structure of statements and queries, rather than on the visualization of the domain objects which are rendered as atomic icons. The expressive power of V is at least equal to that of full first-order predicate logic, containing relations, universal and existential quantification, negation, conjunction, and disjunction.

The graphical user interfaces based on the ER model of the information system let the users compose their own queries. By the means of such an interface, the user accesses and queries any part of the database. On the contrary, the question assistant offers a collection of parameterized question templates that cover pieces of the ER model. There is no way the question assistant can handle user queries targeted to the parts of the ER model not covered by appropriate question templates. The power of the question assistant is limited by the completeness of its collection of question templates. Still, we can hardly imagine the interface of a Web-based database where the user is asked to navigate through its ER model.

7.4 Ask Jeeves

Ask Jeeves[35] is the first successful large-scale question-answering system on the Web. It strives to become a natural language based entry to the entire Web – a smart alternative to existing keyword-based search engines. Ask Jeeves uses a proprietary technology which is not published. Nonetheless, several sources of information make the author of this thesis believe that the question assistant presented in this thesis and Ask Jeeves are closely related systems.

"Automated FAQ Answering: Continued Experience with Shallow Language Understanding" – the first paper in Part 3 of this thesis – was presented at the AAAI Fall Symposium on Question Answering Systems held November 5-7, 1999 in North Falmouth, Massachusetts, USA. According to some experts, Prioritized Keyword Matching resembled the keyword matching technique used by Ask Jeeves.

The following pieces of information, which were available on the Ask Jeeves' web-site during the winter 2000 (they are not available anymore), reveal the common features of Ask Jeeves and the question assistant.

> "Ask Jeeves uses proprietary natural language technology to match your question to one of thousands of 'question templates' and millions of researched links in its knowledgebase. The Ask Jeeves knowledgebase is built by humans who understand what people need when they have questions. Ask Jeeves is continually expanding and becoming smarter over time as people ask more questions."

Like Ask Jeeves, our question assistant also has a number of manually created question templates.

> "The Ask Jeeves knowledgebase consists of three parts: question templates, smart lists, and answer templates. A question template contains the text of basic and alternative forms of a question. Each question template has an associated answer template that contains links to pages on a site that answers a particular query. Smart lists allow one question template to link to several answers."

The reader may recognize the question and answer templates, as well as the clusters of auxiliary question templates that match alternative forms of one user question.

> "Examples of question templates are 'Why is the sky blue?' and 'Where can I find a map of [Name of City]?'. In the first example, there is only one answer link matched to the question, in the second example, there are thousands (one for each city)."

---

[35] http://www.askjeeves.com/, valid in November 2001

Obviously, the question templates of Ask Jeeves contain entity slots, although the author of this thesis has never found any explicit reference to entity or concept slots for Ask Jeeves.

Ask Jeeves and the author of this thesis did their work almost concurrently (with Ask Jeeves slightly ahead) but independently. Ask Jeeves is a company which has developed a proprietary technology, whereas this is a post-graduate research accompanied with a number of publications. Despite the similarities, the author of this thesis denies any possible allegations in plagiarism.

There are also differences between Ask Jeeves and our question assistant. Ask Jeeves tries to cover with question templates the entire Web, whereas our question assistant is bound to the ER model of a particular database. It is not clear whether or not Ask Jeeves uses conceptual modeling of any knowledge domain in order to develop its collection of question templates.

As a result of private conversations at the 1999 AAAI Fall Symposium on Question Answering Systems, the author of this thesis assumed that Ask Jeeves used no analogs of forbidden keywords at that time.

8 Further Research and Conclusions

There are a number of possible directions of developing the ideas implemented in the question assistant. The following subsections discuss them.

8.1 General Issues

Model of Template-Based Question Answering

Automated FAQ answering, discussed in Parts 2 and 3 of this thesis, as well as question answering based on the information system's ER model, employed by the question assistant, both are techniques of template-based question answering. They have common features. Part 5 of this thesis presents a generic model of template-based question answering and its features.

Language Other than English

Making the question assistant "understand" a language other than English, Swedish for example, is an important direction of further development of the question assistant. In Swedish, compound words are considerably more common than in English. One can freely create new words. In order to make the question assistant work with Swedish, we have to teach the system how to split a compound word into its constituents. There has been relevant research done at the Department of Numerical Analysis and Computer Science (NADA) at the Royal Institute of Technology in Stockholm. For example, the spelling checker [Kann et al. 1998] splits compound Swedish words with rather high reliability. The issue is to integrate a software component used in the spelling checker with the question assistant.

Object-Oriented Databases

Object-oriented database systems, "the most promising technology for the next generation of DBMS" [Bertino and Martino 1993], support data whose structure and relationships with other data cannot be mapped directly onto the tabular structure of relational databases. It should not be difficult to adjust the question assistant for question answering using an object-oriented target database because of the analogy between an entity in an ER model and a class of objects. Although the question assistant uses SQL and relational databases, its question-answering technique is based on the properties of ER models of information systems rather than the properties of relational databases.

8.2 XML Output

The output formatting routines are embedded in the source code of the question assistant. There are facilities that allow tailoring the output, such as answer and answer-not-found templates. Still, the customizability of these templates is limited and heavily dependent on the implementation of the `<answer_clause>` and

<answer_src> tags. If we want to change the format of the output produced by these tags, we have to modify the source code and recompile the question assistant.

The question assistant is intended to be a "plug-in" to an arbitrary relational database. Because most database applications have their own graphical user interface, the question assistant could benefit from separation between the question processing kernel and the user output formatting routines. The kernel could produce output as XML (eXtensible Markup Language) [Bray et al. 2000] tagged text, and the parent application's own user interface could use the tags to interpret the text and format the output. For this purpose, we need to develop a standard set of tags and a communication protocol.

8.3 ER Models

Automated Extraction of Question Templates

Would it be possible to automate extraction of the initial question templates from an ER model considering the fact that a collection of question templates mirrors the ER model? [Chen 1983] suggests a number of rules how to construct an ER model by reading its natural language description. [Dalianis 1992] describes a natural language generator as a support tool for conceptual modeling. The user selects the question type from a menu, selects an object on a graphical conceptual schema, and the tool prints natural language statements about the object. Possibly, a similar tool can generate question templates.

Relation between the Complexity of the ER Model and the Number of Question Templates

It requires an empirical investigation to find out how the number of question templates, Q, depends on the number of entities in the information system's ER model, E. We have a hypothesis that, in the majority of business knowledge domains, the number of questions that people ask about a real world concept is limited. In an ER model, there are a limited number of relationships attached to an entity, and there are a limited number of relationships considered in a question. Therefore, for the majority of business knowledge domains, $Q \leq cE$. The constant c depends on the knowledge domain.

This kind of knowledge would allow planning the human resources needed to set up an instance of the question assistant once the complexity of the ER model is given.

8.4 Enhanced Language Analysis

Actor and Action Keywords

Required, optional, and forbidden keywords denote the importance of certain words in the question template. What happens if we regroup these keywords so that they

indicate the actors, actions, and the receivers of the actions in the question template? Would it make the question assistant more "intelligent"?

Open and Closed Word Classes in Information Retrieval vs. Stop-Words in Prioritized Keyword Matching

In Information Retrieval, we distinguish open and closed word classes. Open word classes – nouns, verbs, adjectives, and adverbs – expand by new words as the language evolves. Closed word classes - prepositions (e.g., "on", "by"), determiners (e.g., "a", "the", "some"), pronouns (e.g., "he", "she"), conjunctions (e.g., "and", "or"), auxiliary verbs (e.g., "is", "can", "may"), particles (e.g., "up", "down"), numerals ("two", "three") – are unlikely to acquire new words.

In Prioritized Keyword Matching, the list of stop-words contains closed class words and those open class words that are very common within the subject of the question-answering system. Can we consider *all* adjectives and adverbs as members of the stop-list if they are not listed among required keywords?

Quality of Retrieval of Data Instances

Measurements of recall and precision of the retrieval of data instances, taken during the first test case, have showed the values 0.82 and 0.83. Higher recall is desired. Furthermore, the question assistant has shown not that good rejection – the ability not to retrieve garbage if there are no relevant data instances.

Section 3.1.4 puts forward a hypothesis that it could be possible to create a collection of 20-30 built-in entity-specific text-parsing rules that could fit the majority of the names of real world objects in different business applications. More empirical results are needed to confirm or deny this hypothesis.

Repository of Irregular Forms

In Section 3.2.2, a few not yet answered questions were posed:

- How domain-specific is the repository of irregular forms?
- If we include ordinary language synonyms in a cluster, how deep synonymy do we consider?
- Similarly, if we switch between related nouns, verbs, and adjectives, how deep relations do we consider?
- Where lies the optimal edge between introducing knowledge-domain-specific synonymy in the repository of irregular forms and data-instance-specific synonymy in x-rules? How do we optimize the trade-offs?
- Would the performance of the system be worse if we use an ordinary lexicon, such as WordNet[36]. [Hull 1996] reports that there are problems with using

[36] http://www.cogsci.princeton.edu/~wn/, valid in November 2001

linguistic approaches for stemming in Information Retrieval. Many decisions about the root form of a word, which are properly motivated from the linguistic perspective, are not optimal for Information retrieval performance. Linguistic analysis tools must be tailored for Information Retrieval applications and preferably optimized for the particular domain. And the question assistant is not even a typical Information Retrieval application.

Further research and empirical investigations could answer these questions.

8.5 Conclusions of Part 4 of this Thesis

The *contribution* of Part 4 of this thesis is a new approach to building a question-answering interface – the question assistant – for an information system based on the entity-relationship data model. The main scientific contributions include (i) description of the data structures and algorithms, (ii) estimation of the expected execution time of the algorithms, (iii) evaluation of the performance of the system, as well as (iv) comparison with related techniques and systems. Besides, we discuss the requirements for the system and its maintenance issues.

The question assistant has evolved from the FAQ (Frequently Asked Question) answering systems introduced in Parts 2 and 3. It uses a number of question templates – "frequently asked questions" – extracted from the entity-relationship model of the target database (i.e., the database whose interface the question assistant is). Question templates describe the entities and their relationships as natural language questions. Each question template has an associated answer template.

Unlike static FAQ entries, question templates contain free entity slots, which represent entities in the ER model, to be replaced with data instances. Entity slots are the main innovation that distinguishes the question assistant from its ancestors – the FAQ answering systems.

The question assistant matches the question templates to a user question by means of the Prioritized Keywords matching technique.

Answering a user question takes five steps:

1. The question assistant identifies the data instances and their corresponding entities represented in the user question.

   The system uses word stems (in fact, truncated words that often happen to be the stems) in order to match the words in the user question to the words in data instances ignoring inflections. The repository of irregular forms helps to deal with irregular forms if words.

2. The question assistant retrieves one or several, if any, question templates that are (i) semantically close to the user question and (ii) reference the same entities as the user question.

3. The question assistant substitutes entity slots in the retrieved question templates with the identified data instances, and thus acquires a number of interpretations of the original user question.
4. The user selects the desired interpretation.
5. The question assistant processes the answer template associated with the selected interpretation and answers the interpretation.

The question assistant poses three important *requirements for its target database*. First, the system was developed as a question-answering interface for a structured database. The system does *not* work with a collection of free text documents.

Second, people refer to object by their names, therefore the target database is expected to have columns that contain text of names of people, addresses, titles of events, etc. that are, in most cases, short pieces of text.

Third, the target database should have consistent structure of the text of data instances in the same entity, if possible. For example, human names in one entity should have the same format.

The following *features are specific* for the particular knowledge domain and the target database:

- the question templates and the corresponding entity definitions mirror the ER model of the particular information system;
- the stop-list, used by the Prioritized Keyword Matching algorithm, includes words that are too common for the particular knowledge domain, particular information system.

The other features of the question assistant – the question-answering approach, the data structure, and the algorithms – are *general* for the databases that satisfy the requirements stated in Section 2.1.

The *recall* (ability to retrieve all the relevant data items) and *precision* (ability to retrieve only relevant data items) of the retrieval of data instances are 0.82 and 0.83. The recall and precision of the retrieval of question templates are 0.98 and 0.93. The ability of the system not to retrieve garbage if it lack information to answer the user question is not that good.

What is the competitive edge of the question assistant? The system does not yield better quality of the service than the traditional approaches of Natural Language Processing based on semantic analysis of user queries and subsequent generation of database queries. The competitive edge of the question assistant is its simplicity while rendering still acceptable quality of the service. Administration of the question assistant does not require rare skills: knowledge of conceptual modeling and database design, understanding of the subject area of the system, as well as good command of English are sufficient. The question assistant is designed for

organizations that do not want to use complex and expensive systems.

PART 5

MODEL OF TEMPLATE-BASED QUESTION ANSWERING

1 Introduction

This is the last part of the thesis which formulates an abstract model of template-based question answering as a summary and generalization of the features common for the FAQ answering systems (see Parts 2 and 3 of this thesis), the question-answering interface for relational databases (see Part 4), and possibly a number of similar systems coming in the future.

The AAAI Fall Symposium on Question Answering Systems [Chaudhri and Fikes 1999], held November 5-7, 1999 in North Falmouth, Massachusetts, USA, was an event important for this research. The conference had a specific focus on automated question answering as opposed to more general natural language processing. Practitioners were especially welcome. During the opening speech, the co-chair Vinay K. Chaudhri talked about different approaches and, among others, mentioned template-based question answering. The concept "template-based question answering" had first appeared within the framework of this research. At that time, the concept had not much public content attached to it. Template-based question answering was attributed mostly to Ask Jeeves[37] – a company that operated and never published its own proprietary technology.

The paper "Automated FAQ Answering: Continued Experience with Shallow Language Understanding" (included in Part 3 of this thesis), which describes automated FAQ answering using Prioritized Keyword Matching, was presented at the conference. The idea of question templates having variable parameters (see Section 1.1 in Part 4) was formulated shortly before the conference. The more generic notion of template-based question answering, acquired at the conference, allowed bringing the FAQ entries and the question templates under a common umbrella.

Let us remind ourselves the main features of both kinds of the systems presented previously in this thesis.

1.1 FAQ Answering System

In an FAQ collection, the information supplier tries to answer in advance typical questions that the information users may have. Traditionally such a collection is organized as an ordinary list, which makes it difficult to navigate through the FAQs if the list is too long or chaotic. An FAQ answering system receives a natural language user question and retrieves FAQ entries that match the question.

An FAQ answering system plays several roles in the community of its users:

[37] http://www.askjeeves.com/, valid in November 2001

- *Form of organizational memory.* An FAQ answering system collects past experiences relevant in the community of its users: an FAQ identifies a typical problem, and the FAQ answer presents a solution to the problem. It is an answer garden (a system with such name is described in [Ackerman and Malone 1990; Ackerman 1994]) that helps to improve an organization's memory by providing a database of answers to commonly asked questions.
- *Means of information acquisition.* A natural language based user interface lets people formulate and submit their problems as questions to the system that records and answers these questions. The system collects explicit problem statements which could be difficult to elicit by other means.
- *Means of information dissemination.* The system retrieves FAQs and their answers from its database upon requests expressed in natural human language.

This research is devoted primarily to the last role of an FAQ answering system – means of automated information dissemination.

The FAQ answering system developed by this research answers user questions as follows. When the user submits a question, the system performs exhaustive search through its FAQ collection and matches the question to the FAQ entries (Figure 1.1) using the Prioritized Keyword Matching technique.

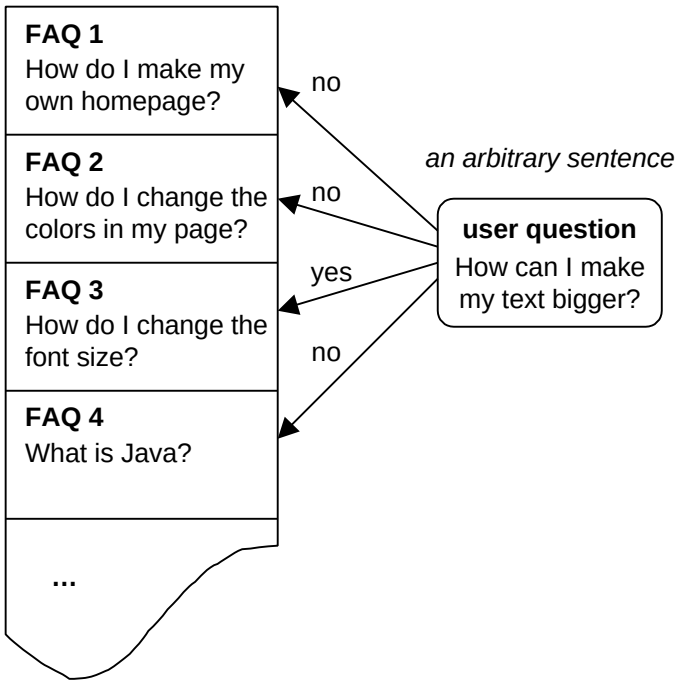

Figure 1.1 System matches the submitted user question to pre-stored FAQ entries, the context of FAQs is HTML.

The idea of Prioritized Keyword Matching is based on the assumption that there are three main types of words in a sentence within a certain context in a certain subject:

- *Required keywords* are the words that convey the essence of the sentence. They cannot be ignored.
- *Optional keywords* help to convey the meaning of the sentence but can be omitted without changing the essence of the sentence. The nuances may change though.
- *"Irrelevant" words*, like "a", "the", "is", etc., are words that are too common in ordinary language or in the subject. The meaning of "irrelevant" words is close to that of stop-words in Information Retrieval. The only difference is that stop-words are assumed always unimportant in a given collection of documents, whereas any of the "irrelevant" words here may suddenly become relevant if used in order to emphasis nuances in a particular sentence in a given collection of sentences. The latter rarely happens.

Each FAQ entry has its required and optional keywords that distinguish the question. There is also the forth type of words – *forbidden keywords*. These are words whose possible presence in a sentence is not compatible with the existing meaning of the sentence. Forbidden keywords are scarcely assigned to FAQs only to distinguish two similar FAQs when the means of required and optional keywords is not sufficient. Forbidden keywords emphasize the difference between both FAQs.

The system has one list of "irrelevant" words common for all FAQ entries but specific for the particular knowledge domain.

Each keyword is represented by a number of synonyms. Word stems (more precisely, truncated words that often happen to be word stems) may be used to include inflections of words.

A notable feature of the Prioritized Keyword Matching technique is that it requires almost no processing of the submitted user question. The system uses a special algorithm that matches required, optional, and forbidden keywords of each FAQ to the submitted user question and determines whether or not the FAQ is semantically close to the user question.

1.2 Question-Answering Interface Based on the Information System's Entity-Relationship Model

The above FAQ answering system is considerably simpler than its NLP (Natural Language Processing) counterparts because it does not comprehend user queries. Nonetheless, it is effective only in domains where concepts and their relationships have rather few data instances. No one is likely to create a separate FAQ entry for each instance in a large data collection.

We can, however, benefit from the simplicity of automated FAQ answering in the task of creating a question-answering interface for a relational database. Let us

introduce the notion of a question template. A question template is a question having entity slots – free space for data instances that represent the main concepts of the question. If we substitute the entity slots with data instances, we obtain an ordinary question.

Creating of an interpretation of the user question has two stages:

1. In the target database (i.e., the database whose question-answering interface the system is), the system locates the data instances that are referred to in the user question (Figure 1.2). The system uses special algorithms to match data instances to a user question.
2. In the repository of question templates, the system (i) finds the templates – "frequently asked questions" – that match the user question, (ii) substitutes their entity slots with the data instances retrieved at the previous stage, and thus (iii) obtains interpretations of the user question (Figure 1.3). Because the question templates are similar to the FAQ entries described above, the system uses the Prioritized Keyword Matching technique to match a question template to a user question.

The question template's "answer" is created by help of a database query template – a formal database query having entity slots for data instances, primarily primary keys. After the slots are filled, the template becomes an ordinary executable query. For example,

```
SELECT start FROM timetable
WHERE timetable.eventid IN
( SELECT eventid FROM event
  WHERE event.placeid = <place-primary-key>
  AND event.eventid IN
  ( SELECT eventid FROM performer
    WHERE performer.performerid = <performer-primary-key>
  )
)
```

could be the query template associated with "Has <performer> performances in <place>?" The database retrieves the time of the performances, if any performance takes place.

The entity-relationship (ER) diagrammatic technique [Chen 1976] is a graphic way of displaying real world concepts, their relationships, and attributes. A written natural language sentence is another way of displaying real world concepts, their relationships, and attributes. A question template, which embodies a natural language question, describes relationships between two or more entities, or between entities and their attributes. A reasonably complete collection of question templates covers the ER model of the target database and can match the majority of on-topic user questions.

The described question-answering interface is a descendant of the FAQ answering system. It inherits two of the three roles of an FAQ answering system: means of information acquisition and dissemination.

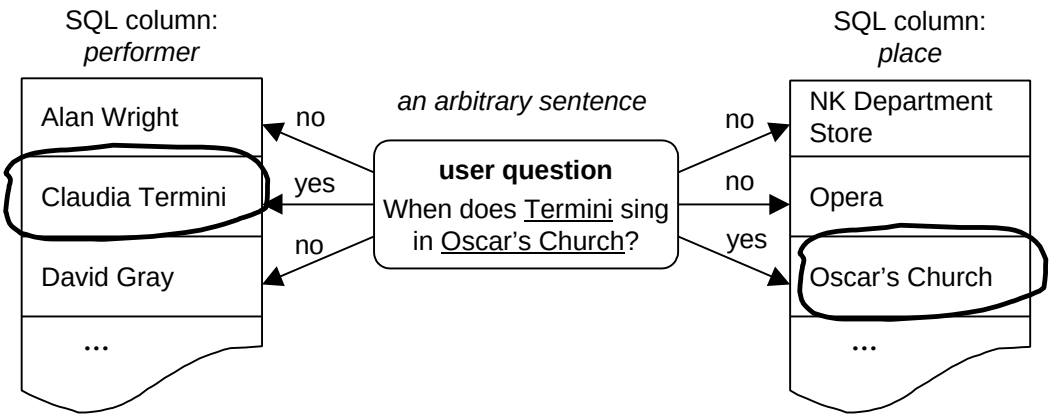

Figure 1.2 Matching a user question to data instances in the target database.

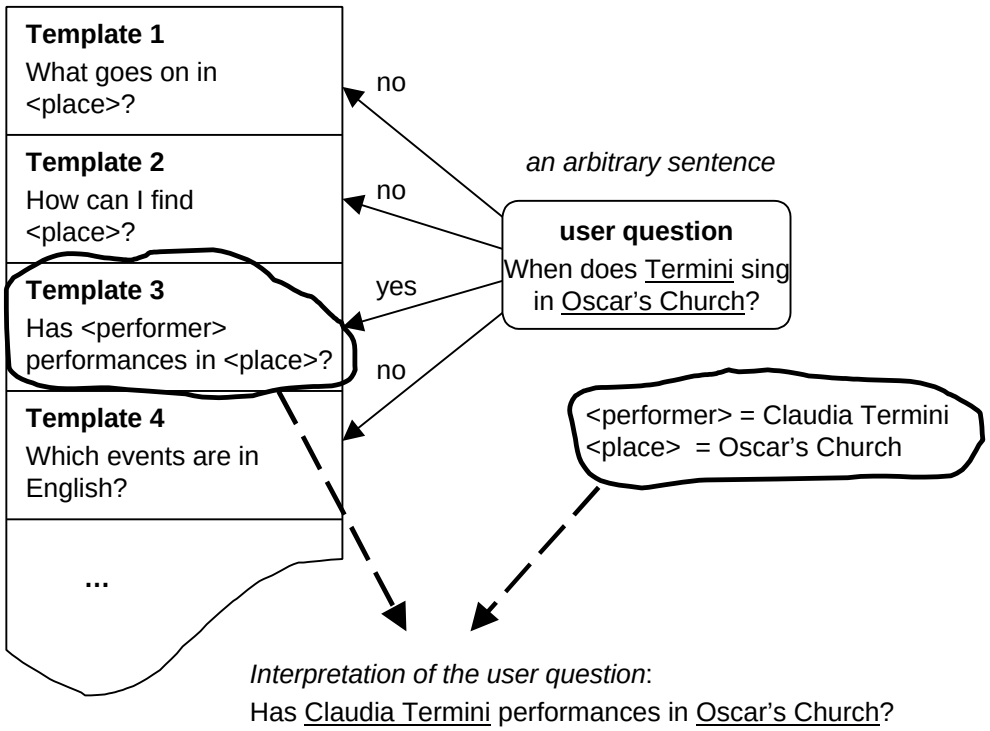

Figure 1.3 Matching the user question to question templates and obtaining an interpretation.

1.3 Structure and Contributions of Part 5 of this Thesis

The main contribution of Part 5 of this thesis is the abstract model of template-based question answering as a summary and brief analysis of the common features

of the FAQ answering system and the question-answering interface of a relational database, both reintroduced above. The result of the analysis is an abstract model of template-based question answering. Section 2 presents the model which focuses on the rationale behind the approach, basic concepts, processes, as well as the advantages and disadvantages of template-based question answering. Section 3 simulates “frequently asked questions”, it gives an insight into the approach of template-based question answering by proposing questions and giving the answers. Section 4 suggests the further research and presents the conclusions of Part 5 of this thesis.

2 Generic Model of Template-Based Question Answering

The ideas of template-based question answering did not appear at once, they were incrementally developed between summer 1997 and summer 2001. Until autumn 1999, the development was conducted without considering the FAQ answering system as a part of a broader branch of question-answering systems. Today, having enough experience at hand, the research puts forward an abstract model of template-based question answering that covers FAQ answering systems and a class of question answering interfaces of structured databases.

The model comprises two views on the approach of template-based question answering. The first view shows the model in the context of a knowledge domain: how the knowledge is transformed into question templates, how and why question templates acquire their main components and features. The second view shows the constituents of the template-based question-answering approach – the general components of the system and the user query.

2.1 Transforming Knowledge into Question Templates

Figure 2.1 shows the chains of transformations that convert a knowledge domain into question templates.

2.1.1 Steps and Conditions of the Transformation

Conceptual modeling [Loucopoulos and Zicari 1992] is an activity that involves eliciting concepts, their attributes, relationships, and restrictions from a fuzzy knowledge domain. While the computer cannot query a fuzzy knowledge domain, it can query the conceptual model. User questions that query a knowledge domain refer to its concepts, their attributes, and relationships (designated by the letters “C”, “A”, and “R” in Figure 2.1). The fundamentals of template-based question answering are matching the concepts, attributes, and relationships in a user question to their counterparts in the conceptual model of the knowledge domain.

Technically, matching a user question to the conceptual model is done by matching the user question to a number of question templates that cover the conceptual model. [Nijssen 1977] explains how the conceptual model describes a set of elementary sentences that carry the information about the knowledge domain. Combining the elementary sentences into intelligible questions results in a set of question templates.

In Figure 2.1, *abstract question templates* are the Nijssen’s elementary sentences combined into questions. Abstract question templates contain concept slots – generic concepts or specific instances. The templates express relationships between concept slots or between concept slots and their attributes. The abstract question

templates mirror pieces of the conceptual model. In a question answering system, they are implemented as either FAQ entries or question templates with entity slots.

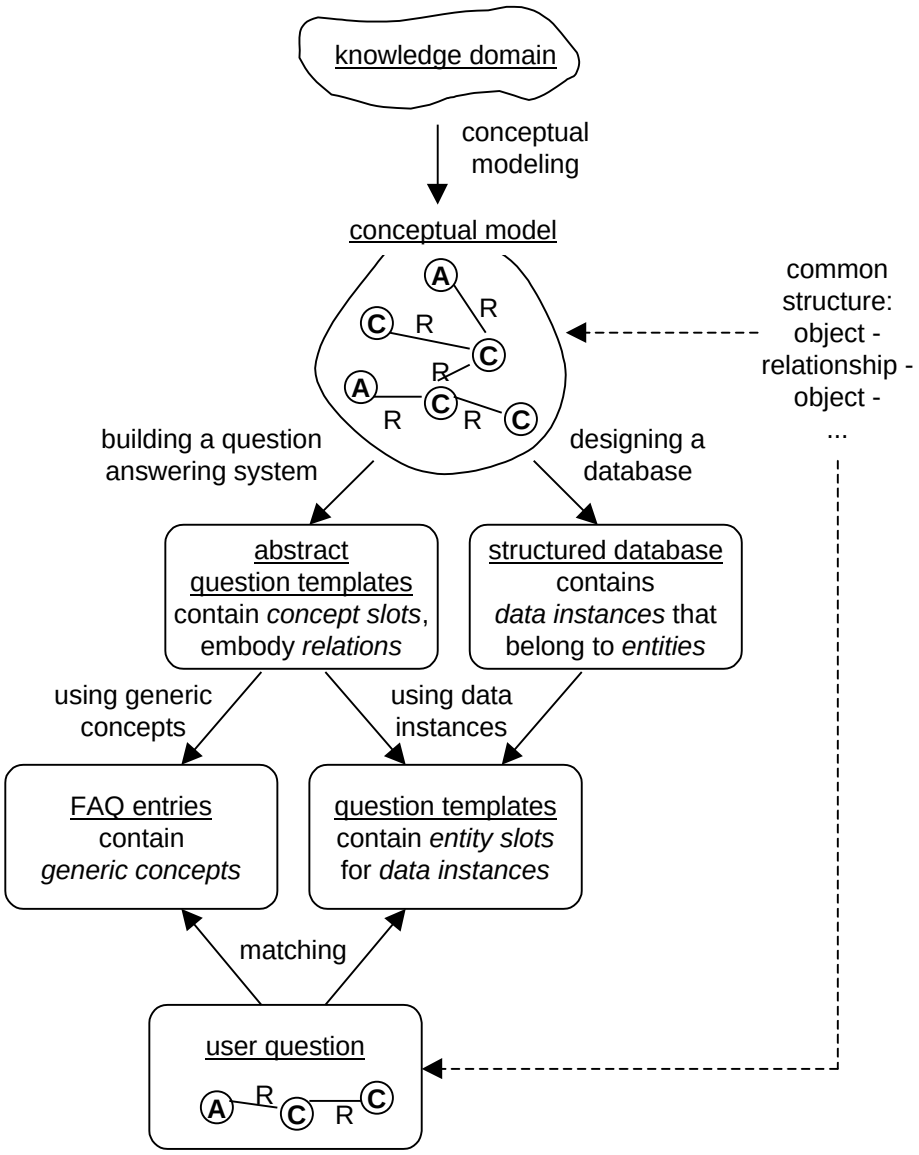

Figure 2.1 Transforming knowledge domain into question templates in order to enable matching of user questions to the knowledge domain.

FAQ entries express questions about generic concepts. An FAQ entry implies static relationships between static concepts or between the concepts and their attributes. We create FAQ entries in order to cover knowledge domains where the concepts have rather few instances.

Question templates with entity slots express questions about specific instances of concepts. Such a template implies static relationships between nonstatic entity slots or between the entity slots and their attributes. An entity slot embodies a variety of instances of a concept. We use such question templates together with a structured

database (e.g., relational database) for knowledge domains where the concepts have many instances.

2.1.2 View of Information Systems Development vs. View of Language Processing

Figure 2.1 presents a view on template-based question answering as an extension of conceptual modeling towards building a question-answering interface for a knowledge domain. The picture is probably not interesting for linguists because, although it deals with natural language interfaces, it does not focus on properties of human language. We do, however, consider the fact that most natural language questions aimed at querying a knowledge domain include objects – generic concepts or specific instances – and their attributes, and relationships between them.

Figure 2.1 presents an approach to building a question-answering interface from the point of view of the information carrier. It focuses on the issue "we have information, how can we better serve it" as opposed to traditional NLP which focuses on "we have a query, how can we answer it". Our question-answering system knows answers before any user questions are asked. The answers are bound to the conceptual model and data in the target database. Therefore "knowledge domain" is placed on the top of the picture whereas "user question" lies at the bottom.

There has been a lot of research done on how to make information systems' interfaces more user friendly (there exists a study field called Human-Computer Interaction). Noticeably less attention is paid to the issue how to make the interfaces more maintainer-friendly. Whatever system we build, we have to think about who is going to maintain it. Eva Ejerhed, the chief scientist at Hapax[38], expressed an opinion that database researchers and linguists traditionally have had difficulties to cooperate [Tångring 2001, p. 34]. There may come up "cultural problems" that may reach even certain "hostility" between both communities. This is a weak point of natural language interfaces. Because many developers and maintainers of information systems do not understand linguistic approaches, they do not use them. Consequently, today's information systems tend not to have natural language interfaces. The time will show whether or not our template-based question answering appeals to these people. But at least we are aware of the problem.

The above considerations explain why Prioritized Keyword Matching, which is developed by this research, requires almost no processing of user questions.

[38] http://www.hapax.com/, valid in November 2001

2.2 Constituents of Template-Based Question Answering

As it was stated in Section 2.1.1, the fundamentals of template-based question answering is matching the objects and relationships in a user question to the concepts, their attributes, and relationships in the conceptual model of the knowledge domain. In order to enable this, the conceptual model is covered by a number of question templates. Hence, the two key constituents of our question-answering approach are user question and question template. Figure 2.2 shows the conceptual model of template-based question answering (not to be confused with the conceptual model of the knowledge domain of the information system where the question-answering interface is applied) – the main concepts and their relationships that take part in the question answering process.

2.2.1 User Questions

The user submits a one-sentence-long *written question* while having an *implied question* in mind. An implied question may have alternatives forms embodied in different written questions. Likewise, a written question may represent different implied questions. For example, a simple "What's up?" in the context of the database on the events in Stockholm may have two meanings: "Hi, how are you doing!" and "What events take place in Stockholm this weekend?"

2.2.2 Question Templates

Question templates express relationships between *concept slots* – generic concepts or specific instances – or between concept slots and their attributes ("question template" in Figure 2.2 represents "abstract question templates" in Figure 2.1). The features of the concept slots and subsequently the question templates are determined by the number of instances of each concept.

Concepts Having Few Instances

For knowledge domains where concepts have rather few instances, we replace the concept slots with *generic concepts* and get "frequently asked questions". For every new concept, relationship, or attribute we create a separate FAQ entry (yet we can combine several concepts, relationships, and attributes in one entry as long as the question remains intelligible).

Concepts Having Many Instances

For knowledge domains where concepts have many instances, we replace the concept slots with *entity slots*, where an entity slot embodies a variety of instances of a concept. We use this kind of question templates together with a structured (e.g., relational) database, where entity slots represent entities in the ER model, and instances of concepts are data instances in the database.

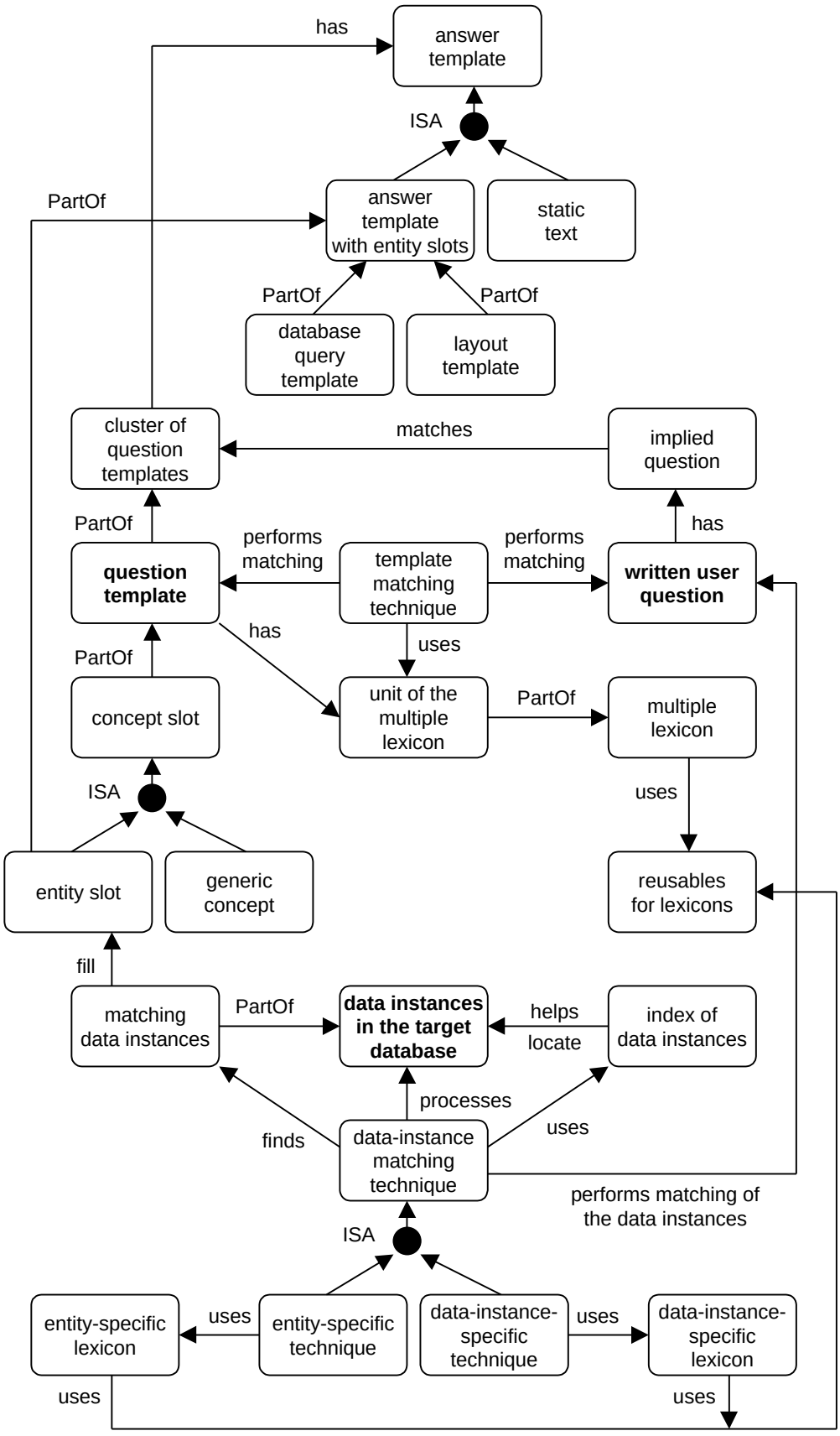

Figure 2.2 Constituents of template-based question answering.

In certain knowledge domains time is important. For example, all scheduled events have temporal aspect. For such knowledge domains, many question templates are expected to have entity slots that deal with time. There must be default time values in order to match user questions where time is implied but not explicitly expressed in the written question.

We can view a question template as a predicate having variable and fixed parameters, where the fixed parameters are entity slots:

$$\exists\ \text{variable}_1, \text{variable}_2, \ldots \text{variable}_n:$$
$$R\ (\text{fixed}_1, \text{fixed}_2, \ldots \text{fixed}_m, \text{variable}_1, \text{variable}_2, \ldots \text{variable}_n)$$

The predicate *R* implements relationships between the parameters. If the question has an answer, the value of the above logical statement is "true". During the question answering process, the values of the fixed parameters – fixed_1, fixed_2, … fixed_m – are bound to the user query, whereas the values of the variable parameters – variable_1, variable_2, … variable_n – are to be found. In order to answer the user query, the system finds all combinations of the variable parameters that retain the value of *R* "true". People familiar with the programming language Prolog may notice that this resembles programming in Prolog.

One question template serves a large number of data instances that pertain to its entity slots. If, however, we want to add a new relationship or attribute, we have to create a new question template (unless we transform the relationships and attributes into meta-concepts and treat the relationships and attributes as instances of the meta-concepts). Each combination of relationships between the concept slots has its own template. Although concept slots in a template may contain variable values, the relationships between them are always static. This is a significant feature of template-based question answering.

Clusters of Question Templates

Question templates are joined into *clusters*. The templates in one cluster represent alternative forms of one question. Hence, a cluster matches one implied question. Technically this means that the template matching technique manages to match one of the cluster's question templates to the written user question.

If question templates have entity slots and, therefore, represent predicates, all the templates in one cluster represent the same predicate.

2.2.3 Answer Templates

Each cluster of question templates has an *answer template*. The features of answer templates depend on what kind of concept slots is used in the corresponding question templates.

Concepts Having Few Instances

The answer template associated with an FAQ entry – a question template that includes only generic concepts – is a piece of *static text*.

Concepts Having Many Instances

The *answer template associated with a question template having entity slots* comprises two components: a database query template and a layout template. The *database query template* turns into an executable database query after its entity slots are filled with data instances. Database query templates are analogous stored procedures on database engines such as Microsoft SQL Server and Borland Database Engine. In terms of a predicate with variable and fixed parameters, the database query template finds the values of the variable parameters after it is given the values of the fixed parameters. The *layout template*, which contains entity slots to be filled with data instances, formats the output.

There is a logical link between a database query template and the corresponding question template: both templates have exactly the same entity slots and, because the relationships expressed in the question template are static, instructions in the database query template are static too. Thus, we skip the difficulty of automatic generation of the database query. Instead, we submit input values to the parameters of the database query template.

2.2.4 Text Matching Techniques

Matching of Question Templates

A *template matching technique* determines (i) whether or not the question in a question template and the submitted user question have a common meaning and (ii) the level of similarity of both sentences if they do have a common meaning. Formally, we can use any sentence matching or language processing technique that satisfies (i) and (ii). The FAQ answering systems and the question assistant developed by this research use Prioritized Keyword Matching.

A significant feature of template-based question answering is use of a *multiple lexicon*, introduced in Section 3.3.1 of Part 2 of this thesis and also discussed in the paper “Automated FAQ Answering: Continued Experience with Shallow Language Understanding” in Part 3. The multiple lexicon solves a long lasting problem of NLP – disambiguation of the meanings of words in the context of a short expression. The lexicon consists of a number of small, analogous, autonomous *units* which operate 5-10, or even fewer, concepts in the context of one question template. On the contrary, the traditional NLP lexicon operates thousands of concepts in the context of the application-specific knowledge domain.

Each question template incorporates one unit of the multiple lexicon. The template matching technique uses the unit when it matches the question template to a written user question. It is the main duty of the unit to “decipher” the meaning of the user

question and to determine the match between the user question and the question template.

The multiple lexicon proposes a refinement of the application-specific knowledge domain. Still, the relationships between certain words that hold in the context of the application-specific knowledge domain need not to be replicated in every unit of the multiple lexicon. The units share *reusables for lexicons*. In Parts 3 and 4 of this thesis, such reusables are called substitutes because the systems apply them as graphical substitutes of pieces of text.

Matching of Data Instances

A *data-instance matching technique* is used only if question templates have entity slots linked to a structured database. We do not use it for automated FAQ answering. The technique matches relevant *data instances residing in the target database* to the written user question and finds *matching data instances* which fill entity slots of the question templates retrieved by the template-matching technique. It would take too long time to match all the data instances in the target database to the user question. With the help of the *index of data instances*, the system quickly selects fewer candidates for more thorough examination.

We should note that the model of template-based question answering does not specify the physical format of the target database. It can be a relational or object-oriented database, or have other database type as long as it has an ER model and is queried using formal queries.

The data-instance matching technique is, in fact, a collection of techniques for matching diverse types of data instances to the written user question. The techniques have two levels of specificity: there are entity-specific and data-instance-specific techniques.

An *entity-specific technique* is applied to data instances that belong to one entity, or a number of analogous entities. For example, different person names have similar features, and a large number of person names are processed according to the same rules. Built-in and custom entity-specific text-parsing rules in Section 3.1.3 in Part 4 of this thesis are entity-specific techniques. The “ruleless” algorithm is the default technique common for a number of entities.

Entity-specific techniques use *entity-specific lexicons* which contain linguistic information specific for data instances that belong to the given entity. For example, linguistic information relevant for processing person names may be not relevant for processing addresses, and vice versa. Entity-specific lexicons share reusables. In the question-assistant, entity-specific text-parsing rules have a common stem dictionary and repository of irregular forms of words.

Data-instance-specific techniques deal with specific single data instances. X-rules in Section 3.1.3 in Part 4 of this thesis are data-instance-specific techniques. Use of

such techniques is limited by the difficulty of treating each data instance separately. Nonetheless, processing on the level of data instances is already used in existing large data collections. For example, when we search the Yahoo directory[39] for "Gordon Sumner", we get the entry for "Sting" which is the pseudonym of the artist.

Data-instance-specific techniques use *data-instance-specific lexicons*. For example, the x-rule in Figure 3.3 in Part 4 of this thesis contains all the information the question assistant needs in order to match "Sting" to user questions. Data-instance-specific lexicons share reusables. In the question-assistant, x-rules may use the same substitutes that required, optional, and forbidden keywords use.

The (i) template matching and (ii) data-instance matching techniques work in different contexts: (i) works in the contexts of question templates, whereas (ii) works in the contexts of entities and individual data instances. Ideally, (i) and (ii) should cooperate because the meanings of words are best to find out in the context of the submitted user question. In the question assistant, there is no explicit cooperation between Prioritized Keyword Matching and text-parsing rules for data instances. Both techniques implicitly interact through the links between question templates and entity slots – certain entity slots belong to certain question templates – as well as by the means of credibility tests (see Section 4.3.1 in Part 4 of this thesis about credibility tests).

Many Small Lexicons

One may object that there are too many lexicons in the model – separate units of the multiple lexicon, a number of entity-specific lexicons, data-instance-specific lexicons. Who is going to keep track of them? The problem is not as big as it seems. The lexicons are small, similar, they share common reusables. Analogous lexicons differ only where the difference is necessary.

2.3 Processes of Template-Based Question Answering

The FAQ answering systems and the question assistant developed by this research answer a user question in three stages:

1. The system receives a user question and matches it to the question templates, as well as the instances of the concepts if there are entity slots rather than generic concepts in the question templates.
2. The system presents its own interpretations of the original question – the matching templates and instances of the concepts – to the user who selects the desired alternative.

[39] http://www.yahoo.com/, valid in November 2001

3. The system retrieves the answer of the user-selected alternative and presents it to the user.

The above stages are the only processes in automated FAQ answering. There are few more processes if the question templates have entity slots and the system must search for data instances in the target database. These processes are described in Section 2.3 of Part 4 of this thesis.

It is difficult to identify accurate processes within the abstract model of template-based question answering without setting the template matching and data-instance matching techniques (see Figure 2.2), because these techniques have their own processes. Perhaps, if we use heavy natural language processing for matching a user question to a question template, we can skip stage 2 – presenting interpretations which approximate the submitted user question. If the system uses machine learning, it must have processes which incorporate it. Currently we can mostly speculate about how different sentence matching and language processing techniques would behave in the settings of template-based question answering.

2.4 Advantages of Template-Based Question Answering

Natural Language Processing and Information Retrieval work within a vague application-specific knowledge domain. This research suggests another approach: we split the application-specific knowledge domain into distinct question-specific knowledge domains. It is easier to process a collection of distinct pieces of a knowledge domain rather than the fuzzy knowledge domain itself.

Narrow Context for Language Processing

Multiple lexicon solves a long lasting problem of NLP – disambiguation of the meanings of words in the context of a short expression. Autonomous units of a multiple lexicon, attached to the question templates, deal with synonymy of the words in the narrow context of one question template rather than in the context of the entire application-specific knowledge domain. Further, entity-specific lexicons and data-instance matching techniques are designed to process the text of data instances in the specific context of the meanings of the entities where the data instances belong.

A question template allows matching similar variations of one written user question. Having a cluster of question templates, we can match as many alternative forms of the same user question as needed.

The system is less error prone: mistakes in one question template or unit of a multiple lexicon do not affect other templates or units of a multiple lexicon, unless the mistakes are made in the reusables.

Maintainability

The knowledge base of the system can be augmented gradually piece by piece. The system is ready to answer questions already with the first question template in the repository.

A unit of a multiple lexicon operates a few concepts in the context of one question. It is easy to maintain each such unit. A multiple lexicon is a finite collection of small autonomous lexicons, therefore it draws rather accurate boundaries between relevant and irrelevant linguistic information with respect to a fixed (although evolving) collection of question templates. With respect to Prioritized Keyword Matching, maintenance of a multiple lexicon is not difficult. We have a hypothesis that the advantages of a multiple lexicon would hold for most sentence matching and language processing techniques: it is easier to maintain a finite collection of small distinct lexicons rather than one lexicon that operates thousands of concepts within a broad knowledge domain with no distinct boundaries between relevant and irrelevant information.

Reusables of a multiple lexicon help to avoid redundancy: several autonomous units share the same reusables. Nonetheless, they decrease the autonomy of the separate units of a multiple lexicon.

The FAQ answering systems and the question assistant presented in this thesis are an empirical proof of maintainability of systems that use template-based question answering. Within the framework of his post-graduate studies, the author of this thesis alone designed the data structures and algorithms, programmed the software, set up and maintained the EKD FAQ answering system (see Part 2 of this thesis) and the question assistant for the SIS database (see Part 4). The HTML FAQ answering system (see Part 3) was set up and maintained together with a MSc student.

Possibility to Use Simple Sentence Matching Techniques

From the very beginning of this research, template-based question answering was combined with Prioritized Keyword Matching. An apparent advantage of template-based question answering is that it permits use of a simple sentence matching technique – simpler than the techniques of Natural Language Processing – while the system still retains competitive performance.

2.5 Disadvantages of Template-Based Question Answering

The only major disadvantage of template-based question answering is that question templates are crated manually (see Section 3.4).

A minor disadvantage is that template-based question answering, if combined with a simple sentence matching technique such as Prioritized Keyword Matching, does

not provide a natural flow of the dialog that resembles human-human communication. Instead, the system creates its own interpretations of the user question, which approximate the original question, and makes the user confirm or select the interpretations to be answered. This is a trade-off between the simplicity of the sentence matching technique and the naturalness of the dialog.

There exist public NLP systems that sustain natural dialogs, such as Eliza[40]. Eliza is operated in the general common sense knowledge domain, and her dialogs are not particularly advanced.

[40] http://www-ai.ijs.si/eliza/, valid in November 2001

3 Questions and Answers on Template-Based Question Answering

This section simulates frequently asked questions on the subject of template-based question answering. It presents the subject from the point of view of a curious mind.

3.1 Is Template-Based Question Answering a Technique of Information Retrieval?

No, it is not. There are at least two reasons for that. First, template-based question answering may be used in natural language interfaces for structured (e.g., relational) databases, whereas Information Retrieval works with free-text documents only. Second, template-based question answering is designed for processing of one-sentence-long user questions, whereas Information Retrieval deals with documents that contain tens or, better, hundreds of words. Auto-FAQ [Whitehead 1995] has applied an Information Retrieval engine for automated FAQ answering, yet it is not clear how good the quality of the FAQ retrieval was.

Within this research, Prioritized Keyword Matching has been used as the template matching technique (see Figure 2.2). Section 7.1 in Part 4 of this thesis compares Prioritized Keyword Matching with the techniques of Information Retrieval.

Can template-based question answering be used for retrieval of free text documents? Please, read Section 3.6.

3.2 Is Prioritized Keyword Matching a Part of Template-Based Question Answering?

Template-based question answering and Prioritized Keyword Matching are two different things. Template-based question answering is an approach to building natural language interfaces where

1. the conceptual model of the application-specific knowledge domain is covered by a number of question templates, which means that the knowledge domain is refined into a number of question-specific knowledge domains, and
2. the system matches a user question to the question templates, which means that it processes the user question within the context of a number of question-specific knowledge domains.

Prioritized Keyword Matching is a technique used for (2): it matches one user question to one question template. Prioritized Keyword Matching seems difficult to use without question templates, whereas template-based question answering can be used without Prioritized Keyword Matching, given that we have other template matching technique in Figure 2.2.

3.3 Is Template-Based Question Answering Easier than Natural Language Processing?

This research has shown that a system that uses template-based question answering together with Prioritized Keyword Matching is easy to maintain.

Perhaps we should rephrase the question: “Would the template-based approach make easier question answering that uses NLP?” Section 2.4 puts forward a hypothesis that the advantages of a multiple lexicon hold for most sentence matching and language processing techniques because (i) each autonomous unit of the multiple lexicon contains only a few concepts and (ii) the multiple lexicon has rather accurate boundaries between relevant and irrelevant linguistic information with respect to a finite collection of question templates rather than a vague knowledge domain. This hypothesis has not yet been proven for any NLP technique.

3.4 Are Question Templates Always Created Manually?

Yes, they are. While we can try to automatize the process of generating natural language sentences that cover the conceptual model [Dalianis 1992], we cannot automatically generate a multiple lexicon which contains question-specific and application domain-specific linguistic information. To our knowledge, no one has managed to automatically generate an ordinary NLP lexicon for an arbitrary knowledge domain. While this is not done, there are no reasons to believe that one can automatically generate a multiple lexicon.

Intelligent decision making during the process of creating question templates takes human reasoning because today’s technologies cannot communicate the meaning of a natural language sentence from the human mind to the computer without any manual work done at some point of the development of the system itself or its imported components. Despite the achievements in NLP, “it is safe to say that computers that carry on conversations, tell jokes and make sarcastic remarks are still the stuff of science fiction and will remain so for the foreseeable future” [Godby 1997].

There exist tools that assist in the process of creating a multiple lexicon, primarily computerized lexicons. Still, it takes human reasoning to decide which linguistic information from the general lexicon is applicable in the context of each specific question template, and which information has to be added from scratch.

3.5 Are there Other Systems that Use Template-Based Question Answering?

The model of template-based question answering is a novel scientific contribution of this thesis. Still, there exist other systems developed using similar ideas.

Askalot is a Swedish company which offers Internet-based technical solutions for customer and public relations services. Its question-answering facility[41] in Swedish uses clusters of question templates, where each template is characterized by a sequence of groups of synonyms. The system matches such sequences to the submitted user question.

Ask CABE[42] is another question-answering facility. According to personal communication with the author of the system, the development was inspired by [Sneiders 1999-b], included in Part 3 of this thesis, and several other sources.

Salut[43] – a European Union 5th Framework research project which maintains a Web-site about eating disorders – intends to use the FAQ answering approach presented in Parts 2 and 3 of this thesis for automated FAQ answering in English and Swedish, and there are plans for French and Spanish.

Relevance of Ask Jeeves to this research has already been discussed in Section 7.4 of Part 4 of this thesis. Ask Jeeves has inspired visionaries of the future Internet reality. According to [Walsh 1999], very soon the Internet search facilities will deploy question templates as a basis to translate the users' questions into database queries. The natural language questions document in detail the needs and desires of the users. Analysis of the question logs and modifying both the question templates and the information source (i.e., a web-site, database, etc.) leads to a better service.

3.6 What are the Requirements for the Information System whose Question-Answering Interface Uses the Template-Based Approach?

This question does not apply to automated FAQ answering.

We have a hypothesis that template-based question answering can be applied for any knowledge domain that can be split into a finite number of concepts and their relationships. The format of the underlying database (hypothetically) does not matter as long as we can apply formal queries.

The Internet is a knowledge domain *not* appropriate for template-based question answering. Ask Jeeves has attempted to cover the entire Internet with a large number of question templates. Still, it seems that the number of the templates is not satisfactory, and it is not clear whether or not it will ever be.

In this research, template-based question answering has been used for a relational database. Section 2.1 in Part 4 of this thesis presents a number of requirements adjusted for the particular template matching and data-instance matching

[41] http://wwwa.askalot.com/sv/, valid in November 2001

[42] http://www.cabe.org.uk/, valid in November 2001

[43] http://www.salut.nu/, valid in November 2001

techniques (see Figure 2.2) used by the question assistant. The most important requirements are:

- *Textual names.* People ask questions about objects having names, therefore the database is expected to have columns that contain proper names – names of people, addresses, titles of events, etc. – that are, in most cases, short pieces of text.
- *Right formulation in the right category.* In the database, objects are expected to be placed into right categories so that data instances belong to the right entities. The data instances are expected to have formulations – names and titles – suitable for the chosen category.
- *Right formulation for the right attribute.* If a class of objects has different attributes, keep them separate and assign specific values accurately.

Section 5.1 in Part 4 of this thesis puts forward a hypothesis that template-based question answering can use also an object-oriented target database because the principles of the question-answering approach are based on the properties of ER models of information systems rather than the properties of relational databases.

Currently we can only speculate how appropriate template-based question answering is for retrieval of information from a collection of free-text documents given that its knowledge domain contains stable entities and their relationships. If we can apply formal queries to such a collection (as most search engines do), we should be able to introduce query templates. If we can introduce query templates, we should be able to introduce question templates as well. In such a case we can use template-based question answering. We do not know yet how advantageous such a system could be.

3.7 Is Template-Based Question Answering Really Better than other Question-Answering Approaches?

Template-based question answering is *not* a technique of Information Retrieval, and it is not clear whether or not it can be applied for retrieval of free text documents (see Section 3.6).

In automated FAQ answering, the template-based approach is *not* appropriate for FAQ retrieval from large existing collections of FAQs stored in free text files. Use Information Retrieval or NLP within the given knowledge domain for that purpose.

There is a not-yet-proven hypothesis that the template-based approach could make easier question answering that uses NLP (see Section 3.3).

Advantages, disadvantages, and limitations of template-based question answering are covered in Sections 2.4, 2.5, and 3.8.

The very first evaluation of the performance of the question assistant is done in Section 6 in Part 4 of this thesis.

3.8 What are the Limitations of Template-Based Question Answering?

Template-based question answering works only in a knowledge domain where we can create a conceptual model of the knowledge domain. The capabilities of template-bases question answering are limited by the possibility to cover the conceptual model with an exhaustive collection of question templates, which comprise the concepts and their relationships, and by the completeness of the collection of question templates.

Further limitations are determined by the template matching and data-instance matching techniques (see Figure 2.2). To our knowledge, there is no evidence showing that refining of an application-specific knowledge domain into a number of question-specific knowledge domains, as it is done in template-based question answering, would diminish the capabilities of any sentence matching or language processing technique.

Used together with a simple template matching technique, such as Prioritized Keyword Matching, template-based question answering does not provide a natural flow of the dialog that resembles human-to-human communication. Instead, the system creates its own interpretations of the user question, which approximate the original question, and makes the user select or confirm the interpretations to be answered.

At least for the nearest future, template-based question answering (like any natural language interface) is *not* recommended for building the only user interface of a database. A question-answering interface is recommended as an easy to use supplement to a more powerful means of querying the database.

4 Further Research and Conclusions of Part 5 of this Thesis

The model of template-based question answering was developed as a summary of the experiences obtained with two instances of the FAQ answering system and the question-answering interface of a relational database discussed in Parts 2, 3, and 4 of this thesis. The systems use the Prioritized Keyword Matching technique in order to match user questions to question templates. A possible direction of the further research is to investigate how different sentence matching and language processing techniques would behave as the template matching technique (see Figure 2.2). It would be interesting to find out whether or not it is easier to maintain a traditional NLP lexicon transformed into a multiple lexicon. There are too possible reasons for that:

- A unit of a multiple lexicon operates a few concepts in the context of one question. It is easy to maintain each such unit. Reusables of a multiple lexicon help to avoid redundancy: several autonomous units share the same reusables.
- A multiple lexicon has rather accurate boundaries between relevant and irrelevant linguistic information with respect to a fixed (although evolving) collection of question templates. A traditional NLP lexicon, on the contrary, works in the context of a vague application-specific knowledge domain with no clear boundaries.

The main *contribution* of Part 5 of this thesis is the model of template-based question answering. The model is abstract and not fixed to a particular sentence matching or language processing technique. The model perceives template-based question answering as an extension of conceptual modeling (see Figure 2.1). Computers cannot query a fuzzy knowledge domain, but they can query its conceptual model. The fundamental of template-based question answering is matching the concepts, attributes, and relationships in a user question to their counterparts in the conceptual model. Technically it is done by matching the user question to a number of question templates that cover the conceptual model.

Figure 2.2 presents the constituents of the template-based question-answering approach – the general components of the system and the user query. Question template and written user question are the key constituents.

Question templates express relationships between concept slots – generic concepts or specific instances – or between concept slots and their attributes. The features of the concept slots and subsequently question templates depend on the knowledge domain:

- For knowledge domains where concepts have rather few instances, we replace the concept slots with generic concepts and get “frequently asked questions”.

- For knowledge domains where concepts have many instances, we replace the concept slots with entity slots, where an entity slot embodies a variety of instances of a concept. We use this kind of question templates together with a structured (e.g., relational) database, where entity slots represent entities in the ER model, and instances of concepts are data instances in the database.

Each question template has its own unit of a multiple lexicon. Hence, the system matches a user question to the question template within the context of the question-specific knowledge domain rather than much broader application-specific knowledge domain. This permits more precise matching. Question templates are arranged in clusters, where a number of templates in one cluster represent different alternative forms of one question.

Question templates are created manually. While we can try to automatize the process of generating questions that cover the conceptual model, currently we cannot automatically generate a multiple lexicon which contains question-specific and application domain-specific linguistic information. Intelligent decision making during the process of creating question templates takes human reasoning because today's technologies cannot communicate the meaning of a natural language sentence from the human mind to the computer without any manual work done at some point of the development of the system itself or its imported components.

From the very beginning of this research, template-based question answering was combined with Prioritized Keyword Matching. An apparent advantage of template-based question answering is that it permits use of a simple sentence matching technique – simpler than the techniques of Natural Language Processing – while the system still retains competitive performance.

REFERENCES

Ackerman M. and Malone T. (1990) Answer Garden: A Tool for Growing Organizational Memory. *Proceedings of the ACM Conference on Office Information Systems*, Massachusetts Institute of Technology, Cambridge, Mass., Apr. 1990, pp. 31-39.

Ackerman M. (1994) Augmenting the Organizational Memory: A Field Study of Answer Garden. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, Nov. 1994, pp. 243-252. Also available as Technical Report 94-40 at Information and Computer Science Department, University of California, Irvine.

Angelaccio M., Catarci T., and G. Santucci. (1990) QDB*: A graphical query language with recursion. *IEEE Transactions on Software Engineering*, Vol. 16, No. 10, pp. 1150-1163.

Bertino E. and Martino L. (1993) *Object-Oriented Database Systems. Concepts and Architectures.* Addison-Wesley.

Bray T., Paoli J., Sperberg-McQueen M. C., and Maler E. ed. (2000) *Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation 6 October 2000.* URL: http://www.w3.org/TR/REC-xml, valid in July 2001.

Bretan I. (1995) *Natural Language in Model World Interfaces.* Thesis in partial fulfillment of the requirements for the degree of Licentiate of Technology, Dept. of Computer and Systems Sciences, Stockholm University / Royal Institute of Technology, Sweden.

Bubenko J., jr. (1994) Enterprise Modelling. *Ingènierie de Systemes d'Information*, Vol.2, issue 6, p.657-678.

Burg J. F. M. (1997) *Linguistic Instruments in Requirements Engineering.* IOS Press, Netherlands.

Burke R., Hammond K., Kulyukin V., Lytinen S., Tomuro N., and Schoenberg S. (1997-a) Question Answering from Frequently Asked Question Files: Experiences with the FAQ Finder System. *AI Magazine*, Vol. 18, No. 2, pp. 57-66.

Burke R., Hammond K., Kulyukin V., Lytinen S., Tomuro N., and Schoenberg S. (1997-b) *Question Answering from Frequently Asked Question Files: Experiences with the FAQ Finder System.* Technical Report TR-97-05. University of Chicago Computer Science Department.

Cailliau R. (1995-2001) *A Little History of the World Wide Web.* URL: http://www.w3.org/History.html, valid in November 2001.

Carlberger J., Dalianis H., Hassel M., and Knutsson O. (2001) Improving Precision in Information Retrieval for Swedish Using Stemming. *Proceedings of NODALIDA'01 – 13th Nordic Conference on Computational Linguistics*, 21-22 May 2001, Uppsala, Sweden.

Cerf V. (1993) *The birth of the ARPANET.* URL: http://www.internetvalley.com/cerf-how-inet.txt, valid in November 2001.

Chaudhri V. and Fikes R., ed. (1999) *Question Answering Systems. Papers from the 1999 AAAI Fall Symposium.* Technical Report FS-99-02, November 5-7, North Falmouth, Massachusetts, USA, AAAI Press.

Chen P. P. (1976) The entity-relationship model: Towards a unified view of data. *ACM Trans. Database Systems,* Vol. 1, No. 1.

Chen P. P. (1983) English Sentence Structure and Entity-Relationship Diagrams. *Information Science,* Vol. 29, No. 2 and 3.

Cormen T. H., Leiserson C. E., and Rivest R. L. (1990) *Introduction to Algorithms.* MIT Press.

Czejdo B., Elmasri R., Rusinkiewicz M., Embley D. W. (1990) A Graphical Data Manipulation Language for an Extended Entity-Relationship Model. IEEE Computer, Vol. 23, No. 3, IEEE Computer Society Press, Los Alamitos, pp. 26-37.

Dalianis H. (1992) A Method for Validating a Conceptual Model by Natural Language Discourse Generation. *CAISE-92 International Conference on Advanced Information Systems Engineering, Lecture Notes in Computer Science,* No. 593, Springer Verlag, pp. 425-444.

EKD (1998) *EKD User Guide.* Internal document at DSV, Stockholm University / Royal Institute of Technology.

ELEKTRA (1996) *ELEKTRA – Electrical Enterprise Knowledge for Transforming Applications,* ESPRIT Program 7.1, proposal of the project No.22927.

F3 (1994) *From Fuzzy to Formal, Esprit III Project 6612. Requirements Engineering Handbook.* Swedish Institute for Systems Development (SISU).

Fenton N. E. and Pfleeger S. L. (1997) *Software Metrics: A Rigorous and Practical Approach.* Second Edition, International Thomson Computer Press.

Godby J. (1997) *Natural Language Processing at OCLC.* URL: http://orc.rsch.oclc.org:5061/papers/what_is_nlp.html, valid in November 2001.

Gromov G. R. (1995-2001) *The Roads and Crossroads of Internet History.* URL: http://www.internetvalley.com/intval.html, valid in November 2001.

Grosz B. J., Jones K. S., and Webber B. L., ed. (1986) *Reading in Natural Language Processing.* Morgan Kaufmann Publishers, Inc., pp. 545-549, 563-584, 585-593.

Hammond K., Burke R., Martin C., and Lytinen S. (1995) FAQ Finder: a Case-Based Approach to Knowledge Navigation. *Proceedings. The 11th Conference on Artificial Intelligence for Applications,* Los Alamitos, CA, USA, IEEE Comput. Soc. Press, pp. 80-86.

Hearst M. (1999) User Interfaces and Visualization. *Modern Information Retrieval*, ed. Baeza-Yates R. and Ribeiro-Neto B., Addison-Wesley Longman Publishing Company, pp. 257-339.

Huber G. P. (1984) *The nature and design of post-industrial organizations.* Management Science, Vol. 30, No. 8, pp. 928-951.

Hull D. A. (1996) Stemming Algorithms – A Case Study for Detailed Evaluation. *Journal of the American Society for Information Science*, Vol. 47, No. 1, pp. 70-84.

ISC (2001) *Internet Domain Survey.* URL: http://www.isc.org/ds/, valid in November 2001.

Kann V., Domeij R., Hollman J., and Tillenius M. (1998) *Implementation aspects and applications of a spelling correction algorithm.* NADA report TRITA-NA-9813, the Royal Institute of Technology, Stockholm, Sweden. PostScript and PDF documents available also at http://www.nada.kth.se/~viggo/papers.html, valid in November 2001.

Kekäläinen, J. (1999). *The effects of query complexity, expansion and structure on retrieval performance in probabilistic text retrieval.* Ph.D. dissertation, Department of Information Studies, University of Tampere. Acta Universitatis Tamperensis 678. ISBN 951-44-4596-1; ISSN 1455-1616.

Krovetz R. (1993) Viewing Morphology as an Inference Process. *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, New York, pp. 191-202.

Kwok C., Etzioni O., and Weld D. S. (2001) Scaling Question Answering to the Web. *Proceedings of the 10th International World Wide Web Conference (WWW10)*, 1-5 May 2001, Hong Kong.

Larsson K. (1999) *Automatiskt besvarande av frågor ställda på ett naturligt språk.* Thesis in partial fulfillment of the requirements for the degree of Master of Science in Computer and Systems Sciences, Dept. of Computer and Systems Sciences, Stockholm University / Royal Institute of Technology, Sweden (in Swedish).

Loucopoulos P. and Zicari R., ed. (1992) *Conceptual modeling, databases, and CASE: an integrated view of information systems development.* John Wiley & Sons.

Lovins J. B. (1968). Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*, March 1968, Vol. 11, pp. 22-31.

Mearian L. (2001) Schwab taps natural-language search engine. *Computer World*, 16 July 2001, p. 8.

Nijssen G. M. (1977) Current Issues in Conceptual Schema Concepts. *Architecture and Models in Data Base Management Systems. Proceedings of the IFIP Working Conference on Modelling in Data Base Management Systems*, Nice, France, 3-7 January 1977. North-Holland Publishing Company, pp. 31-65.

Popovic M. and Willett P. (1992) The Effectiveness of Stemming for natural-Language Access to Slovene Textual Data. *Journal of the American Society for Information Science,* Vol. 43, No. 5, pp. 384-390.

Porter M. F. (1980) An Algorithm for Suffix Stripping. *Program,* July 1980, Vol. 14, No. 3, pp. 130-137.

Russell S. J., and Norvig P. (1995) *Artificial Intelligence. A Modern Approach.* Prentice-Hall, Inc.

Salton G. and McGill M. J. (1983) *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc.

Salton G. (1989) *Automatic Text Processing. The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley.

Sneiders E. (1998) FAQ Answering on WWW Using Shallow Language Understanding. *Information Systems in the WWW Environment. IFIP TC8/WG8.1 Working Conference*, 15-17 July 1998, Beijing, China, pp. 298-319, Chapman & Hall.

Sneiders E. (1999-a) *Automated FAQ Answering on WWW Using Shallow Language Understanding.* Thesis in partial fulfillment of the requirements for the degree of Licentiate of Technology. Report series No. 99-001, ISSN 1101-8526, Department of Computer and Systems Sciences, Stockholm University / Royal Institute of Technology, Sweden.

Sneiders E. (1999-b) Automated FAQ Answering: Continued Experience with Shallow Language Understanding. *Question Answering Systems. Papers from the 1999 AAAI Fall Symposium.* Technical Report FS-99-02, November 5-7, North Falmouth, Massachusetts, USA, AAAI Press, pp.97-107.

Sneiders E. and Larsson K. (2001) *Application and Maintenance Aspects of an FAQ Answering System.* Report series No. 01-007, ISSN 1101-8526, Department of Computer and Systems Sciences, Stockholm University / Royal Institute of Technology, Sweden.

Tångring J. (2001) Språkkänsla ger smartare sökning. *Datateknik 3.0*, No.2, 8 February 2001 (in Swedish).

Walsh J. P. and Ungson G. R. (1991) Organizational Memory. *Academy of Management Review*, Vol.16, No.1, pp. 57-91.

Walsh B. (1999) Searching for Online Customer Service. *Network Computing*, May 1999, Vol. 10, No. 11.

Whitehead S. D. (1995) Auto-FAQ: an Experiment in Cyberspace Leveraging. *Computer Networks and ISDN Systems*, Vol. 28, No. 1-2, pp. 137-146.

Winiwarter W. (1999) An Adaptive Natural Language Interface Architecture to Access FAQ Knowledge Bases. *Proceedings of the 4th International Conference on Applications of Natural Language to Information Systems*, Vienna, Austria.

Xu J. and Croft W. B. (1998) Corpus-Based Stemming Using Cooccurrence of Word Variants. *ACM Transactions on Information Systems,* Vol. 16, No. 1, pp. 61-81.