Keyphrase Extraction from Single Documents in the Open Domain Exploiting Linguistic and Statistical Methods

Alexander Thorsten Schutz

Submitted in fulfillment of the requirements for the degree of
Masters of Applied Science M.App.Sc

SUPERVISOR:
Dr. Siegfried Handschuh

EXTERNAL EXAMINER:
Dr. Diana Maynard,
University of Sheffield

DEAN OF SCIENCE:
Dr. Gerry Morgan

DIRECTOR:
Prof. Dr. Stefan Decker

Digital Enterprise Research Institute, National University of Ireland, Galway
Date of original submission: August 21^{st}, 2008
Date of revised submission: October 30^{th}, 2008

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost I would like to express my gratitude to Dr. Siegfried Handschuh, for bringing me on board 2 years ago, and for his supervision over the course of this undertaking.
Thanks also go to Prof. Dr. Stefan Decker and Prof. Dr. Manfred Hauswirth for providing an excellent and stimulating research environment.

Many people have actively contributed to the completion of this thesis, and I am grateful to every single one of them – unfortunately this list is hopelessly incomplete: The anonymous judges who dedicated their time during evaluation – this thesis would not be the same without their data. VinhTuan Thai, Tudor Groza, and Laura Josan for reading chapters and drafts, and providing valuable feedback – it was all well received. Aidan Hogan, Fergal Monaghan and Gearoid Hynes who offered to read drafts and helped out with their judgement and competence on a number of paragraphs that were hard to formulate. Christoph Lange who I could consult to improve my sloppy mathematical notation. Conor Hayes, Brian Wall and John Breslin for sorting out something. Milena Caires, for putting my head right at times...I reckon the past months weren't always enjoyable. Bettina Friedrich, Mark Hillebrand, Marian Möhren, Arndt Paschke, Torben Reuter, and others of the old Saarbrücken gang, who kept checking up on me every once in a while and who were up for a chat to see how things were going.
Last, I would like to thank my parents Heidi and Manfred Schutz for being supportive in every step I took, which ultimately led me here.

The work presented in this thesis was supported (in part) by the European project NEPOMUK No FP6-027705 and (in part) by the Lion project supported by Science Foundation Ireland under Grant No. SFI/02/CE1/I131.

Declaration

The work presented here was undertaken within the Digital Enterprise Research Intitute, National University of Ireland, Galway.
I confirm that no part of this work has previously been submitted for a degree at this or any other institution and that it is my original work, unless otherwise stated.

Galway, October 2008, Alexander Thorsten Schutz

Date of original submission: August 21^{st}, 2008
Date of revised submission: October 30^{th}, 2008

Abstract

In digital libraries, *keyphrases* are an important instrument for cataloguing and information retrieval purposes. In literature research, they provide a high-level and concise summary of the content of textual documents or the topics covered therein, allowing humans to quickly decide whether a given text is relevant. As the amount of textual content on desktops grows fast, keyphrases can contribute to manage large amounts of textual information, for instance by marking up important sections in documents, for example, to provide increased user experience in document exploration. However, only few documents, whether authored by the user or retrieved from digital libraries on the internet, have keyphrases assigned as embedded metadata description, although vendors have enabled the relevant file formats with structures capable of storing such information.

The most popular solution to automatic keyphrase extraction from textual documents KEA, which utilises machine learning techniques, is not very accessible. Although available as free software component, it requires a training step on a number of documents, thereby imposing a burden on potential users.

As keyphrases are a description of textual data, the consideration of Natural Language Processing (NLP) tools in order to automate the extraction process is obvious. The goal of this thesis is to demonstrate how linguistic and statistical methods can be combined to perform automatic keyphrase extraction from textual data, putting the emphasis on single documents. The proposed software component does not need a training step, can be utilised off the shelf, and is applicable for English, German and French. It is evaluated quantitatively on a medium-sized corpus with a priori assigned keyphrases, whereas a user study gives insight into the acceptance of the algorithms' results in a practical setting. Evaluation results show that the approach is comparable with the current state-of-the-art, while potential for performance improvement still exists.

Preface

If you're dull as a napkin, don't sigh;
Make your name as a "deep" sort of guy.
You just have to crib, see,
Any old book by Kripke
And publish in AAAI.

A hacker who studied ontology
Was famed for his sense of frivolity.
When his program inferred
That Clyde ISA Bird
He blamed – not his code – but zoology!

If your thesis is utterly vacuous,
Use first-order predicate calculus.
With sufficient formality,
The sheerest banality
Will be hailed by all as miraculous.

If your thesis is quite indefensible,
Reach for semantics intensional.
Over Montague grammar,
Your committee will stammer,
Not admitting it's incomprehensible!

[Henry Kautz]

Chapter 1

Introduction

> *Here is Edward Bear, coming downstairs now, bump, bump, bump, on the back of his head, behind Christopher Robin. It is, as far as he knows, the only way of coming downstairs, but sometimes he feels that there really is another way, if only he could stop bumping for a moment and think of it.*
>
> A. A. Milne, Winnie the Pooh, 1926

A number of years ago, back in the early 1990's, the value and importance of keywords – and to be more precise, *keyphrases* – was undeniable. Literature research was barely possible without a mixture of freely formed phrasal queries and adhering to the established controlled vocabularies providing domain-specific index terms for publications and books. The world wide web was still in its infancy, and search engines only had begun to index web pages. A lot of information did exist, but it required knowledge of two kinds to retrieve it: *where* and *how*.

The change came with the digital revolution. Cheap storage space, increasing processing power and matured database systems were the technologies that enabled the creation and maintenance of large data warehouses. Network access and efficient full text indexes greatly contributed to exposing information in a unprecedented manner. Formerly used primarily as a communication medium, the internet had become an information retrieval instrument. Home pages were created, and the world wide web started growing at an exponential rate. When Altavista started to get lost in its index, Google arrived and the first hit usually was acceptable. Keyphrases, that once were so important to identify relevant content, seemed to become redundant as people started to rely on the new convenience that came along with full text search. However, if anyone thought that by the end of the 1990's the information explosion would have

reached its peak, or that thanks to Google any information was only one click away, he had no clue what was about to begin – things had only just started ...

A few years later, Google had re-invented itself, and transformed from a pure facilitator to an information provider and creator, meanwhile indexing books and making a large amount of literature electronically available. Traditional publishers, who were controlling the market before the internet age started to lose ground as the scientific community was seeking for a more direct channel to disseminate its research. The Web2.0 movement enabled easy content creation and distribution on the fly for everybody. The wheel of information creation just kept spinning faster and faster.

At the same time, broadband technology had exceeded the critical mass, and in many households, personal computers resembled entertainment centre and information hub at the same time, accumulating and reproducing textual content on the local hard drive. Many applications on the web and the desktop started to provide an interface between both worlds, effectively blending the two into each other, and increasing the amount of data available even more.

With the growth of information, when searching for content, simple full text search began to reach its limitations, as now, a single word would fit into the many contexts existing on the web, and therefore, on the desktop.

What has been lacking over the past years is a method to enrich available documents of textual content with short, but precise phrases describing the topics covered therein. This kind of information – if available beyond a critical mass – could enable progress and solutions for those lost in data: search engines could provide a more precise result set for a given query, desktop users could organise and retrieve their documents by content and not by file-system folders, and recommendations could base their algorithms on an additional layer of high quality content-related meta-data. All of a sudden, *keyphrases* which were thought to be buried in libraries forever, started to become relevant again.

1.1 Motivation

In (digital) libraries, *keyphrases* are an important instrument for cataloguing and information retrieval purposes. For someone skimming through a large collection of textual documents, they provide a high-level, concise description of the content of each document or the topics covered therein, allowing him to decide whether it is relevant or not. However, a greater number of application scenarios exist, as the nature of the functionality is relatively generic. For instance, keyphrases can be utilised in automatic summarisation tasks, as containing sentences can be extracted and composed to a

more natural, prosaic summary of a document in question. In information retrieval, an additional index of keyphrases could be exploited besides the common TFxIDF measure to provide document similarity and clustering, thereby yielding more precise results over single keyword-based approaches, as the latter ones are more vulnerable to *ambiguity*.

As the amount of textual content on desktops grows fast, keyphrases can contribute to manage large amounts of textual information, for instance, by marking up important sections in documents to provide increased user experience in document exploration. However, as will be shown later on, only a very limited number of documents, whether authored by the user or retrieved from digital libraries on the internet, have keyphrases assigned as metadata description, although vendors have enabled the relevant file formats with structures capable of storing such information. With the emerging trend to include *semantic web technologies* on the desktop, the so-called *semantic desktop*, a first step is to lift metadata from various file formats into an explicit form, mostly by simply accessing embedded vendor or format-specific metadata fields. Unfortunately, if no content-related metadata for texutal documents is embedded, little can be lifted. If content-related metadata was existent, embedded in documents and easily retrievable, semantic technologies could reach their full potential by exploiting such free-form metadata as a first step, for example, by efficiently and automatically selecting an appropriate domain ontology based on the topics mentioned in the keyphrases, providing meaningful, machine-readable and interoperable semantics.

1.2 Goal & Scope

In many libraries, keyphrases and index terms are still assigned manually, ensuring a high standard of quality, at the expense of time and labour. From a librarian's point of view, this path is perfectly reasonable, as a quality of service has to be guaranteed. On the desktop, however, not providing easily accessible tools facilitating the (semi)automatic extraction of keyphrase candidates for textual documents has led to the situation we face today: metadata is actually being mined for use in a number of scenarios, but for the ever growing mass of textual data, it is non-existent, or still implicit, and very hard to retrieve.

Existing machine learning solutions to keyphrase extraction rely on a training step, thereby imposing an additional burden to potential users. As keyphrases are a description of textual data, the consideration of NLP tools in order to automate the extraction

process is obvious. While shallow NLP techniques are a long way from language understanding, in combination with statistical processing they still can be helpful in many ways, providing a first stop in content-metadata extraction, which then can be used as input for more sophisticated technologies.

The desktop scenario for keyphrase extraction poses a number of unique challenges. Not everyone speaks English, and while it is infeasible to provide a solution for every language possible, the conceived solution should be extendable to a wider variety of languages with a reasonable amount of effort.

In information retrieval, NLP solutions often carry a stigma of being slow and unreliable. Typically, as a user is sitting in front of the desktop, waiting for in- and output, it is important for any application with user interaction to operate within reasonable runtime parameters, which are acceptable to the user. Therefore, the approach needs to be able to scale to typical desktop systems, making it possible to provide live and online processing.

As the extraction process is fully automated, the extraction algorithm is likely to make mistakes, most obviously resulting in a lack of precision due to spuriously predicted keyphrase candidates. Hence, it is also a challenge to keep the amount of error to a minimum while still retaining the best possible recall/coverage, to spare the user a tedious selection process in a subsequent interaction where he has to confirm or reject the proposed candidates.

Although overlapping in some points, and sharing similarities in the techniques utilised, the approach discussed in this thesis is not concerned with the tasks carried out in classical Information Extraction (IE) or Automatic Term Extraction (ATE).

The goal of this thesis is twofold: Firstly, it will be demonstrated how linguistic and statistical methods can be combined to perform automatic keyphrase extraction from textual data, putting the emphasis on single documents. A quantitative evaluation will be conducted to assess an objective performance measure for the approach, whereas a user study will give insight into the acceptance of the algorithms' results in a more practical setting.

Although NLP techniques have contributed to academic and industrial niche solutions in certain domains [1], the semantic web community has recognised NLP as being capable of improving the knowledge acquisition bottleneck only to a limited extent [2].

[1] for example, by providing natural language interfaces to databases [55]

[2] prominent examples would be information extraction components developed within the KIM platform [56] or question answering in the AquaLog undertaking [43]

Therefore, it is my personal intent to present how freely available (shallow linguistic) resources can be composed to a fast, robust, flexible and multilingual NLP middleware that – if exposed as service on the desktop or uniquely combined with an application – will be able to automate the tedious process of manual keyphrase assignment to textual documents on a daily basis. As a result, it will be made easier to explicitly enrich textual resources with such metadata, thereby creating the foundation for semantic web and semantic desktop approaches which rely on rich metadata, in order to effectively support computer users, for instance, with intelligent document recommendation or expert finding algorithms.

The outcome of this thesis will be a freely-available software component facilitating the keyphrase extraction process in a real world scenario – the desktop of the user – off the shelf, at the same time being able to reliably process real-world data in common document formats.

1.3 Structure

The following chapter introduces background knowledge in content metadata acquisition, natural language processing, and semantic desktop concepts, situating the thesis into the context at the interface of metadata extraction for knowledge based systems where pragmatic solutions from research are applied in practice, before it concludes with a summary of important related and relevant work. Chapter 3 analyses use-cases and depicts in detail the considerations underlying the approach, giving a high-level overview of the design decisions that were made and the preprocessing that is necessary, whereas chapter 4 continues to describe components and aspects specific to implementation. Subsequently, chapter 5 reports on two experiments that were conducted to assess the quality of the implemented tool. The results of a medium scale quantitative evaluation and of a user study with 47 subjects are being discussed, before chapter 6 summarises the main contribution of this thesis, speculates on future work and use-cases and finally concludes with closing remarks.

Chapter 2

Background

The aim of this thesis is to propose a solution for the automatic extraction of keyphrases from single documents, which is – in some aspects – very much related to approaches in Automatic Term Extraction (ATE), and thus, also to collocation extraction. The extracted keyphrases are intended to be used as a first-stop, free-form metadata description for documents, and therefore, the first part of this chapter will introduce the general idea of content-related metadata from textual data and its role in the semantic web age, and ultimately lead to a sketch of the semantic desktop project NEPOMUK, providing a real-world scenario as a setting for the thesis. Subsequently, as the underlying data sources comprise natural language, the enabling NLP techniques and technologies for content-related metadata acquisition from text will be discussed. Due to the similar and overlapping nature of keyphrase extraction and ATE, the last part of this chapter will survey relevant state-of-the-art approaches in both areas.

2.1 Content-Related Metadata on the Desktop

Metadata on contemporary desktop systems exists for many resources and can take several forms. The Free On-line Dictionary of Computing [1] gives the following definition:

meta-data *data*

> (Or "meta data") Data about data. In data processing, meta-data is definitional data that provides information about or documentation of other data managed within an application or environment. For example, meta-data would document

[1] meta-data. (n.d.). The Free On-line Dictionary of Computing. From Dictionary.com website: `http://dictionary.reference.com/browse/meta-data` – retrieved 2008-08-10

> data about data elements or attributes, (name, size, data type, etc) and data about records or data structures (length, fields, columns, etc) and data about data (where it is located, how it is associated, ownership, etc.). Meta-data may include descriptive information about the context, quality and condition, or characteristics of the data.

The Oxford Digital Library elaborates on **types of metadata** [2]:

descriptive metadata: information describing the intellectual content of the object, such as MARC cataloguing records, finding aids or similar schemes

administrative metadata: information necessary to allow a repository to manage the object: this can include information on how it was scanned, its storage format etc (often called *technical metadata*), copyright and licensing information, and information necessary for the long-term preservation of the digital objects (preservation metadata)

structural metadata: information that ties each object to others to make up logical units (for example, information that relates individual images of pages from a book to the others that make up the book itself)

On the desktop, a number of tools exist to extract and provide *technical metadata* from those proprietary formats and binary data, such that the metadata modelling in terms of the standards proposed by the semantic web movement (RDF, etc.) can be performed seamlessly.

However, the acquisition of so-called *descriptive* or *content-related metadata* from resources is still considered problematic, as this sort of information is highly subjective to the individual. This activity usually involves a human in the loop, who manually provides such information. For example, in the scenario of personal photo collections, someone will annotate a picture with the persons appearing on it if they are known to the annotator. For textual resources, someone will annotate a document with content-metadata if it is considered pertinent, such as title, authorship information or index terms. At a later point, this information can be used to retrieve resources enriched in such a way more easily, as the annotator already is aware of the association between the resource in question and its annotations. Manually creating this form of metadata has been considered a burden in the past, as it is a tedious and laborious process demanding discipline and diligence. However, assisted assignment of metadata – when

[2] `http://www.odl.ox.ac.uk/metadata.htm` – retrieved 2008-08-09

automated to some extent, such that individuals are given the option to choose from a selected set of recommendations – becomes acceptable, as examples in the Web2.0 world (e.g., collaborative tagging systems) have shown.

2.1.1 Content-Related Metadata

The vast majority of user-centric data, particularly on the desktop computer of the average user, is encoded in natural language. For a long time already, vendors have recognised the need to represent content-related metadata descriptions embedded into documents: Adobe has reserved metadata fields for PDF documents (*title*, *author*, *subject*, *keywords*), Microsoft supports a similar set of metadata fields in MS Word documents. The OpenDocument format, and OASIS standardisation effort in general, also propose such a set of content-related metadata descriptions to be integrated into their document formats. Furthermore, document engineering efforts have been working on enriching documents with embedded ontological metadata descriptions based on suitable domain ontologies, as for instance proposed by Groza et al [32].

Adoption by the Mainstream

Content-related metadata has been recognised as a crucial piece of information by librarians, although in reality it has barely been embedded into electronic document formats, neither on the desktop nor by online publishers.

A corpus study of computer science research articles freely available for download suggests that much of the scientific literature in this domain is lacking even the most simple form of embedded content metadata [3]. Of 1,298 PDF documents, only 244 contained embedded *authorship* and *title* metadata, and only 16 documents (1.125%) specified embedded *keyword* information.

These findings give evidence that, despite the fact that the most common document formats on the desktop – PDF, Microsoft Word and OpenDocument Format – already specify metadata fields for such purposes, they are, to a large extent, ignored by the users. So far embedded metadata has been of little use, not adding incentive for the users to manually enrich mentioned metadata fields, which can be a time-consuming and tedious task, requiring a high level of discipline and diligence. Also, relying on fulltext search has become a convenient method for information retrieval in general, be it on the web, in libraries or on the desktop. However, with the increasing amount of

[3] The corpus was kindly provided by Tudor Groza and Ioana Hulpus, and comprises CIKM, ESWC, ISWC, WWW, EKAW, K-CAP, DEXA Proceedings within the years 2004 and 2007, and an additional number of workshops

digital information, fulltext search will necessarily decrease in precision, as witnessed since the beginning years of Google when the common credo was "if you don't find it on the first result page, there is no use skimming through the rest of them". Much has changed since then, and often it is necessary to skim through additional result pages Google returns for some query. With growing indexes of the search engines, the length of the search queries had to grow as well to include terms providing additional context to the query, in order to discriminate the result set: effectively, search queries have become keyphrases.

Back to the desktop and the documents typically encountered there, along with their (non-existing) metadata: the *author* field is occasionally being populated by values initially supplied by the licensee of the authoring tool (Microsoft Word/OpenOffice), other content related fields such as *title*, *subject* and *keywords* are mostly left blank, which is a bad scenario in an era where metadata is mined and lifted from application specific structures into an *open, interoperable format* that has been proposed by the *semantic web* endeavour and pushed forward by the *Web2.0* movement. The slowly emerging success of the *semantic web*, including the Resource Description Framework (RDF) [21], and with it, the advance of *open, interlinked data* [4] is not only changing the way of information access on the internet, it also is **currently** changing the landscape of desktop computing, towards the *semantic desktop*.

2.1.2 The Social Semantic Desktop

As broadband technology has become reality for many computer users, data available on the web and on the desktop are merged, streamed over the internet and continuously updated, as experienced with *content syndication* like *RSS* and *ATOM*.

With the advance of peer-to-peer (P2P) technology, which – besides sharing content – has also successfully been applied in metadata exchange [74], the vision of the *social semantic desktop* [20, 62] has become within close reach, towards a *richly interconnected platform*, playing the part of metadata provider **and** consumer alike.

The semantic web paradigm, that – in order to be interoperable – metadata needs to conform to open, shared conceptualisations expressed by *ontologies* [34], explicitly stating the *semantics* of the domain in question, also holds for the *social semantic desktop*. If information is lifted and expressed explicitly in a unified way, applications would be able to look beyond their own data model, and integrate their data with

[4] `http://www.w3.org/2008/Talks/0617-lod-tbl/` – retrieved 2008-08-08 — the *Linked Open Data* initiative has been named "potentially world changing" by Tim Berners Lee, the inventor of the World Wide Web and visionary of the Semantic Web

data yielded from other applications, or even different desktops. The social semantic desktop helps users to organise, classify and interlink user data, not only on a file level, but also on a more fine-grained *resource* level, such that emails, address book entries, pictures, locations, etc., can be interrelated in a meaningful way, and shared among users in a social context if the need arises. However, since data on the desktop (usually) has a more personal character than shared data on the web, the underlying mechanisms and ontologies on the semantic desktop should reflect this difference, allowing for easily adjustable vocabularies and schemas adhering to personal mental models on an individual basis. As the work described here has been carried out as part of the EU project NEPOMUK [5], which aims at providing a standardised description and reference implementation of the *social semantic desktop*, a brief overview of the proposed architecture, the available knowledge structures and some selected applications therein will be given next, whereas further specifics of the use-case within NEPOMUK are outlined in section 3.1.3.

The NEPOMUK Reference Implementation of the Social Semantic Desktop

The goal of the NEPOMUK project is threefold: (a) to formulate a standardised description of a social semantic desktop architecture, (b) to realise its reference implementation, and (c) to evaluate a number of use cases. In the following, (a) and (b) will be covered briefly.

The current (intermediate) implementation of NEPOMUK has been designed as a Service Oriented Architecture (SOA) with semantic web paradigms in mind, comprising of three layers [31]: (i) the *presentation layer*, providing user interfaces for applications, (ii) the *semantic middleware*, offering centralised *data services* and the core functionalities of the platform, and eventually, (iii) the *network communication layer*, realising the social aspect by establishing connections with other NEPOMUK systems.

Figure 2.1 [31] illustrates the three-layered architecture with its components, which will be detailed in more depth below.

The *presentation layer* offers user interfaces to functionalities provided by the *semantic middleware layer*, which are used for data presentation, creation and manipulation. They comprise standard interfaces well-known from common desktop systems such as a local *file-browser*, an instant *messaging client*, etc., but also cover *wikis* and

[5] Networked Environment for Personalized, Ontology-based Management of Unified Knowledge

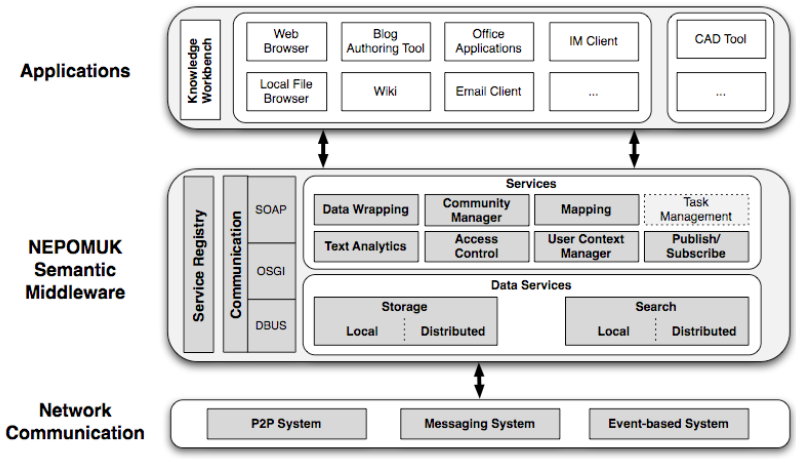

Figure 2.1: The General NEPOMUK Architecture

blog tools. In some cases, plugins for popular commercial products such as MS Outlook have been developed, facilitating seamless integration of email or calendar data resources to be managed by NEPOMUK.

Providing the core services of the platform, the *middleware layer* serves under two aspects: accommodating (meta)data *storage* and *search* facilities (which can be either *local* or *distributed*) and exposing functionalities such as text analytics (which includes the *keyphrase extraction* described in this thesis, *speechact detection* and *semantic tagging and disambiguation*), *access control* and *user profiling*, and others, where for each category mentioned, a number of dedicated services are accessible to programmers via API such that applications for the *presentation layer* can be built if required.

Eventually, the *network communication layer* manages the linking to other NEPOMUK systems in the network, and the relevant communication aspects such as *messaging* and *data exchange* (for distributed search and storage), which is implemented as *peer-to-peer* file and resource sharing system. Figure 2.2 depicts the idea of an interconnected NEPOMUK "universe" in a distributed search scenario.

For more detailed information, see the NEPOMUK deliverables D6.1 [33] [6] and D6.2.A [60] [7].

[6] `http://nepomuk.semanticdesktop.org/xwiki/bin/download/Main1/D6-1/D6.1_v10_NEPOMUK_1st_Version_Backbone.pdf` – retrieved 2008-08-08

[7] `http://nepomuk.semanticdesktop.org/xwiki/bin/download/Main1/D6-2-A/D6.2.A_v10_NEPOMUK_Intermediate_Architecture.pdf` – retrieved 2008-08-08

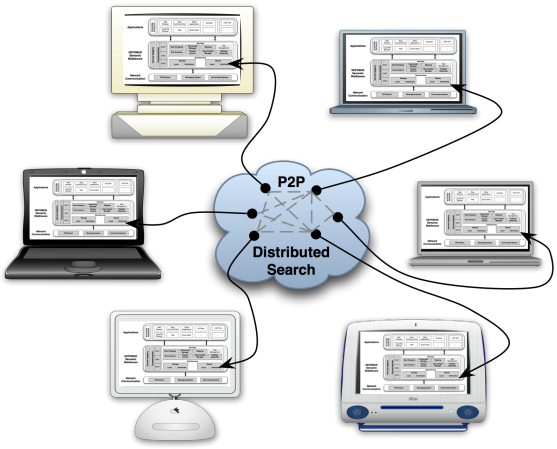

Figure 2.2: Social Semantic Desktops in a Peer-to-Peer Scenario

In order to enable a mutual understanding between different NEPOMUK systems, the concepts and entities typically encountered in a desktop environment have to be modelled explicitly, at the same time allowing for individual customisation of personal mental representations, addressing user-specific needs. Therefore, the ontological knowledge structure in NEPOMUK adheres to a layering of three levels, with each level addressing more specific needs in personal knowledge management. The three levels identified are (i) the *representational level*, (ii) the *upper level*, and (iii) the *lower level*, as depicted in figure 2.3 [60].

At the representational level, the foundations and building blocks for the lower levels are given, in the form of ontology definition languages providing vocabulary that may be used when constructing domain specific ontologies. Besides RDF and RDFS, NEPOMUK relies on the NEPOMUK Representation Language (NRL) [69], making use of named graphs [11] and allowing open and closed world semantics, e.g., for privacy and provenance information. At the upper level, the NEPOMUK Information Elements (NIE) ontology models concepts commonly known from desktop systems – files (images, audio, MS Word documents, etc.), address book entries, calendar data, etc. – integrating vocabulary specified by already existing metadata standards such as ID3, MPEG7, EXIF, and iCalendar, whereas the NEPOMUK Annotation Ontology (NAO) provides conceptual structures for integrating annotations and tags, very much like encountered by the Web2.0 movement. [8]

Near the lower level, the Personal Information Model (PIMO) ontology expresses

[8] see `http://www.semanticdesktop.org/ontologies/` – retrieved 2008-08-09 — for more detailed information on the NEPOMUK ontologies.

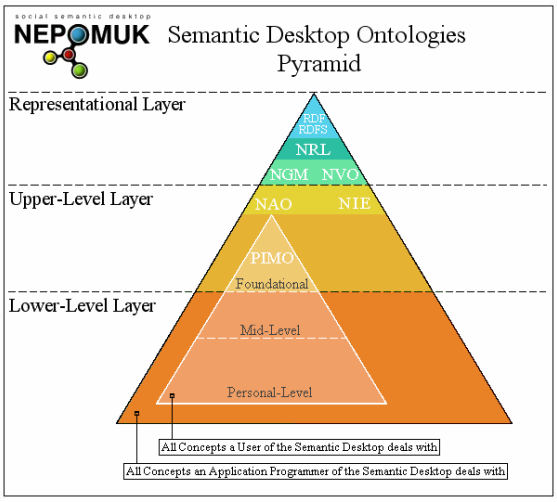

Figure 2.3: Different Levels of NEPOMUK Knowledge Structure

personal information models of individuals. As a consequence, the PIMO has a dynamic touch and can be extended and customised on a case-by-case basis. PIMO has been designed such that the foundations in upper level ontologies prevent inconsistencies to a large extent. In case information modelled by two differently customised PIMOs needs to be interpreted, the *ontology mapping service* in the middleware of the architecture attempts to establish a mapping between the two conceptualisations. For more detailed information about PIMO, see the NEPOMUK PIMO draft deliverable [9].

NEPOMUK has been deployed on a number of systems, as Java implementations based on the Eclipse platform exist for Microsoft Windows, Mac-OS and Linux. However, the most exposed implementation is perhaps NEPOMUK-KDE, which has been bundled officially with the popular graphical user interface KDE [10] for Linux since its 4.0 release early in 2008. Also see the NEPOMUK deliverable D7.2 [11] and the official NEPOMUK-KDE website [12] for more recent information.

2.1.3 Benefits of Keyphrases on the Semantic Desktop

Now that the scenario has been set, a simple but very valid question might pop up:

[9] `http://dev.nepomuk.semanticdesktop.org/wiki/PimoOntology` and `http://dev.nepomuk.semanticdesktop.org/repos/trunk/ontologies/pimo/latex/pimo.pdf` – retrieved 2008-08-09

[10] `http://www.kde.org/` – retrieved 2008-08-09

[11] `http://nepomuk.semanticdesktop.org/xwiki/bin/download/Main1/D7-2/D7.2_v11_NEPOMUK_KDE_Community_Involvement.pdf` – retrieved 2008-08-08

[12] `http://nepomuk.kde.org/` – retrieved 2008-08-08

> *If there are a couple of documents on the desktop, and the domain vocabulary is already in place, why don't we just walk over each document word by word and see which terms are matching in the vocabulary?*

The problems faced here are (i) uncertainty, and (ii) availability. How is it possible to decide which domain vocabulary to settle on, if not known *a priori*?

Ontology libraries like OntoSelect [13] [9] and semantic web search engines such as Swoogle [14] [22] or Watson [15] [19] have indexed a large number of ontologies and offer an interface enabling the retrieval of appropriate schemas that fulfil a given query, for instance, a matching string value of a class name or label.

One way of estimating the best fit could be testing for each distinct (content)word whether ontologies containing concepts (or labels) for the word in question exist. In reality however, this attempt is barely feasible, for two reasons:
(a) given the great number of existing ontologies, it is a huge overhead to query each of them for each (content)word.
(b) the compositionality of natural language increases the magnitude of complexity, for example, the querying for single words would ignore more complex terms.

It might also be perfectly possible that no schema is available for the domain in question (it may be irretrievable at the time, or simply not exist), which would result in a dead end, and no keyphrase generation at all.

The main idea here is to use the keyphrases as a first step to select the appropriate domain vocabulary. The reduced set of keyphrase candidates will provide a less noisy summary of the topics mentioned in a document. This reduced set, in fact, does enable querying mentioned ontology libraries for good-fitting schemas, which then can be retrieved for further semantic annotation *in addition* to the free-form metadata provided by the keyphrases.

Besides, although the primary use-case for this approach to keyphrase extraction has been the semantic desktop, it is obvious that such free-form metadata could be *immediately* very useful to common computer users on the traditional desktop, and a number of use-cases which are equally applicable to the latter are given in chapter 3.1.

As the input for a keyphrase extraction algorithm is data from textual documents, the utilisation of natural language processing techniques is an obvious choice. Thus, in

[13] `http://olp.dfki.de/ontoselect` – retrieved 2008-08-09
[14] `http://swoogle.umbc.edu/` – retrieved 2008-10-22
[15] `http://watson.kmi.open.ac.uk/WatsonWUI/` – retrieved 2008-08-09

the following section, a brief overview of the NLP concepts suitable for this undertaking will be given.

2.2 Natural Language Processing for Multilingual Keyphrase Extraction

As the keyphrase extraction functionality is to be deployed on the desktop, dealing with real-world data, in order to be reliable it has to be prepared to process documents of unrestricted text, varying in size. Short documents however pose a problem for statistical algorithms, which rely on observations, and low frequencies tend to lead to unreliable predictions. Here, a shallow linguistic annotation is exploited for a number of purposes as outlined below, enabling the use of statistical algorithms at a later stage.

2.2.1 Linguistic Pre-Processing & Shallow NLP Components

Often regarded as time and resource savvy, not robust, slow, inflexible, monolithic, academic and too complicated, NLP has failed to make an impact on the modern desktop so far. In the meantime however, shallow linguistic processing can be performed at an acceptable speed on the data commonly encountered in desktop scenarios, providing an output that is potentially better suited for content-based extraction purposes. This section introduces the operations that have commonly been found fundamentally necessary for shallow NLP. The steps covered here provide a foundation for further levels of analysis, including the employment of statistical-based algorithms.

Language Detection

In order to sport the component with support for multiple languages, an identification of the respective language used is inevitable, as linguistic resources later in the process need to be adjusted or changed completely according to the findings of the identification. For instance, a Chinese document could be used as input, but the tokenisation process in Chinese is completely different from Western European languages.

Systems based on *character n-grams*, as proposed by Beesley [4], have been found to be particularly well suited for the task of language identification, as they provide higher certainty over short input data than *word-based systems* that perform lexical look up of characteristic words for a particular language [30].

The application of word-based systems is also problematic when identifying languages such as Chinese, where the process of tokenisation is a difficult one.

Tokenisation & Sentence Splitting

As a text comprises sentence segments, and a sentence is composed of tokens, it is important to split the larger units into their smaller subunits at the correct position. This way, an appropriate linguistic analysis can take place, as for each layer (token-layer, sentence-layer) the set of appropriate linguistic tools may differ. Hence, the first processing step for a linguistic analysis is to divide a given text into *lexical units* (tokens / words) and to detect sentence boundaries.

Part-of-Speech Tagging

Two major categories of part-of-speech-taggers are either probabilistic, using statistical language models (bi-gram, tri-gram, n-gram, Hidden Markov Models (HMMs)) like the tagger Text and Trigrams (TnT) [6], or they rely upon a large base of rules and constraints, like CLAWS [42] or the Brill Tagger [8].

Due to idiosyncratic, language-specific phenomena, most languages have their individual tagset to which part-of-speech taggers provide a mapping, a problem that will be revisited in chapter 3.2.3 and 3.3.

Besides being a fundamental step in any NLP pipeline, part-of-speech tagging in keyphrase and term extraction is important for so-called boundary approaches [5], or approaches applying a linguistic filter to discriminate the candidate set, thereby permitting only a number of n-gram patterns determined by constraints on composition of part-of-speech tags [29, 28, 76].

Stemming / Lemmatisation

In a scenario that ultimately depends on observations, sparse data is always a problem. In the context of NLP, when looking at wordforms, inflections introduce sparseness by encoding information such as tense, number, gender or case. While English is a so-called weakly inflected language with only plural inflection for nouns, and four different types of inflection for regular verbs, in the moderately inflected Germanic and Romance languages a greater variety of inflection can be observed, which ultimately contributes to sparse data. In order to overcome this problem to some extent, a reduction of the words to their *morphological root* or *lemma* (or base form) is usually

performed, such that observations over varying word forms of the *lemma* can be collapsed, and increased frequency counts over the *lemma* can be considered subsequently. For this purpose, two different techniques exist, *lemmatisation* and *stemming*.

Lemmatisation takes the context of a word form into consideration, such as its part-of-speech tag, and may refer to a lexicon for look up of irregular words.

Stemming reduces and modifies the suffix of a given word form to its root according to a list of transformation rules and algorithmic procedures, e.g., by stripping the plural ending "-s" from the word "apples", with no underlying linguistic knowledge applied. Stemming produces a number of invalid words due to errors introduced by over- and understemming, a fact that has been acknowledged but neglected as a trade-off for speed. Two popular stemming algorithms are the Porter [57] stemmer, and the Lovins stemmer [44].

In Information Retrieval (IR), stemming has been preferred over lemmatisation due to its fast runtime behaviour, although in an NLP pipeline, lemmatisation should be considered due to its accuracy in producing valid word types as output.

Noun Chunking

Noun chunking refers to the process of detecting non-recursive and non-overlapping noun phrases in a sentence, while complex decisions about potentially ambiguous prepositional phrases can be made at a later stage. The analysis dividing a text into segments of noun chunks is fairly simple, and often relies only on morphosyntactic categories, expressed as a grammar, or a finite state machine over part-of-speech tags. Therefore, in contrast to parsing (and performing constituency analysis) it can be achieved quickly, as no deeper analysis needs to take place.

Ramshaw and Marcus have identified chunking as a technique potentially useful for index term generation [58], which is also similar to keyphrase extraction.

2.2.2 Statistical Processing

Building statistical models usually starts with counting frequencies over a layer of information, be it tokens, lemmas, part-of-speech tags or head nouns. Statistical language models based purely on absolute frequencies are not very suitable for describing content in language precisely, as the information in language is not equally distributed [16]. For instance, short and frequently occurring function words such as

[16] although the assumption of a random distribution in language has lead to the application of a number of elegant probabilistic models, i.e Bayes

Rank	Word	Frequency	Rank	Word	Frequency
1	the	5776384	51	have	170417
2	of	2789403	52	that	165805
3	and	2421302	⋮		
4	a	1939617	99	years	82878
5	in	1695860	100	way	82343
6	to	1468146	101	our	81997
7	is	892937	⋮		
8	to	845350	1500	excellent	6150
9	was	839964	1501	reality	6148
10	it	834957	1502	winter	6139
12	for	768898	⋮		
13	with	606027	265064	wormhole	3
14	he	605749	⋮		
15	be	603178	322770	undecodable	2
⋮			⋮		
50	what	173582	443839	sphinx	1

[a] Total Types in BNC: 921,043

Table 2.1: Word Frequencies in British National Corpus [a]

prepositions are often used as markers and do not contain much information, an observation that was made by Zipf [79] and later picked up and formalised by Shannon [68] in the information theoretic *noisy channel model* and further discussed by Miller and Newman [51]. As many other natural phenomena alike, language exhibits a so-called Zipfian distribution, meaning that in a sufficiently large corpus, the majority of observations is distributed over only very few words, whereas the majority of words only accumulate very few observations among them. In terms of frequency f and rank r, Zipf's observation can be expressed such that there is some constant k which remains roughly equal as a product of rank and frequency for all words in the corpus: $k = r \cdot f$. In other words, the frequency of a word is inversely proportional to its statistical rank, as table 2.1 and figure 2.4 document for the British National Corpus, where items at the top of the table occur overwhelmingly frequent, whereas at the bottom of the table a very large proportion of low frequent items can be observed.

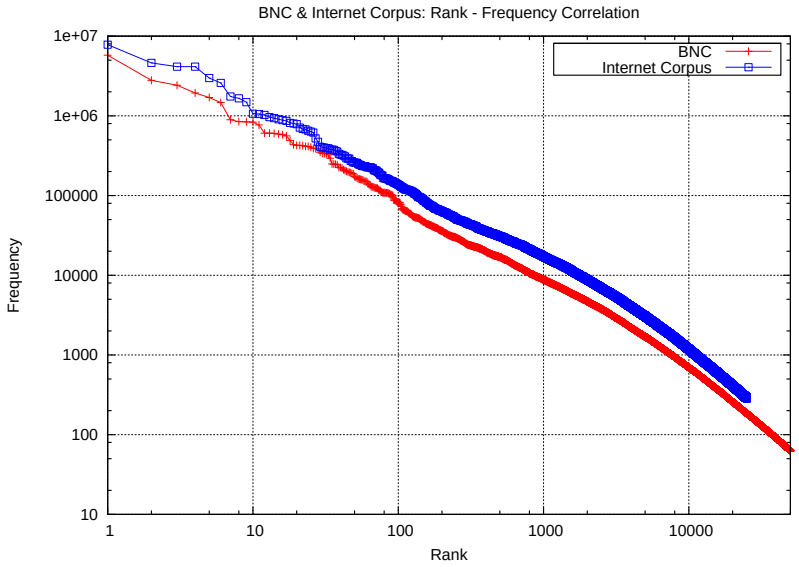

Figure 2.4: Zipf Distribution for BNC and English Internet Corpus

Association Measures, Hypothesis Testing & Information Theory

In statistical NLP, Bayesian models combine the probability of combinations of linguistic features, for instance the probability of the token "rest" being marked up (in simplified part-of-speech terms) either as noun or as verb. Bayesian statistics is the basis for so-called Likelihood Estimations, and has been applied to computing co-occurrence probabilities. Expectations also play an important role in hypothesis testing, which can be applied to ATE, collocation discovery [17] or co-occurrence problems.

However, in most linguistic applications employing hypothesis tests, it is not so much the actual test-value which is important, but rather the fact that those tests provide a ranking which is used to separate interesting occurrences (that is, combinations formed not by chance) from uninteresting ones. Examples of hypothesis testing approaches are the *t-test*, *Pearson's* χ^2 *test* [46](chapter 5) or *log-likelihood* as described by Dunning [23]. Hypothesis testing methods have been used in combination with a balanced corpus in order to determine the specific terms of a domain corpus, as well as measures originating in IR such as Term Frequency / Inverse Document Freqency (TF/IDF). Instead of using hypothesis testing, Church and Hanks demonstrate

[17] A collocation can be described as a unit of combined co-occurring words, where the meaning of the unit carries more than the pure composition of its parts, which is mostly the case for idioms, but also for multi-word expressions. In this context, the null hypothesis proposes that the co-occurrence is merely by chance, and it can be rejected if the result of the hypothesis test exceeds a critical value, which indicates that a collocation was found.

how *mutual information* can be exploited to identify interesting associations between co-occurring words, given a sufficiently large corpus [13].

However, some problems remain with the afore mentioned statistical measures. Low frequent observations at the tail of the Zipf Distribution (so-called *rare events*) and data-sparseness have always been problematic for statistical methods when computing the significance of lexical items for a text. For instance, Pearson's χ^2 measure or the log-likelihood statistic become unreliable when applied to data that has been observed less than 5 times, as Pedersen et al report in [54]. Dunning argues that χ^2 dramatically overestimates the significance of rare events [23], while Kilgarriff points out that in case very frequently occurring events (in the magnitude of 1000s), the whole armada of hypothesis testing relying on a random probability distribution is used inappropriately [40].

In a typical desktop scenario, it is very unlikely that very frequent events will occur to the stage where it poses a problem for the work described here, however, sparse data resulting from short documents are likely, such that appropriate counter measures have to be considered, as further described in chapter 4.1.

2.3 Keyphrase & Term Extraction: State of the Art

At first glance, keyphrase extraction and Automatic Term Extraction (ATE) (also referred to as Automatic Term Recognition (ATR)) seem to deliver largely the same product: relevant terms for textual data. However, some subtle, but important differences exist, and this section intends to point out those differences between the two disciplines, before going on with reviewing relevant work in both fields.

ATE as such is very much concerned with the identification of domain specific (and possibly unknown) terminology from large data collections, whereas keyphrase extraction takes a more document-centric attitude, emphasising the descriptiveness of the extracted phrases with respect to the originating text. Turney [75] notes the difference in specificity between keyphrase extraction and IE, to the point that the output of keyphrase extraction is much less restricted by categories and types than in IE. Given these characterisations, two dimensions can be identified such that keyphrase extraction and ATE are situated at opposite ends of the scales, as figure 2.5 and table 2.2 illustrate.

1. Input Data: ATE operates on large amounts of domain specific data, whereas keyphrase extraction considers a single document as data source.

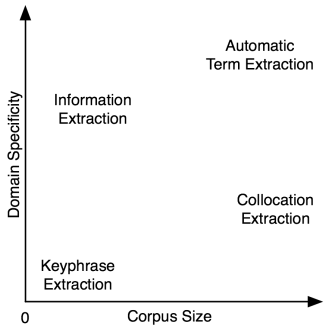

Figure 2.5: Distinctive Dimensions for Keyphrase Extraction, Automatic Term Extraction and Information Extraction

	Corpus Size	Domain Specificity
Keyphrase Extraction	single document	none
Information Extraction	single document	moderately specific
Collocation Extraction	large corpus	barely specific
Term Extraction	large corpus	very specific

Table 2.2: Distinctive Dimensions for Keyphrase Extraction, Automatic Term Extraction and Information Extraction

2. Domain: ATE often is performed around some sort of domain specific vocabulary, and either such vocabulary is used as background knowledge, or the approach is aimed at creating/enriching such knowledge structures.

Additionally, in ATE, online processing becomes unfeasible as a typical user application, whereas keyphrase extraction should provide a result for a requested document in a – for the user – acceptable amount of time, which goes along with the necessarily very large data set ATE is operating on. As Bourigault explains [5],

> *The appearance of a new terminological unit is most often a parallel process to that of the birth of the concept which it represents. This "birth" is marked by the consensus of a certain scientific community. This consensus is attested only when the* occurrences *of this linguistic expression, or term-to-be, shows a stable correlation to the same object in the subject field, uniquely and completely, in the writings of the agents of this scientific community.*

At the same time, a large proportion of methods used in ATE have been found working very well for *collocation testing*, and both fields share a considerable amount of characteristics such as the large quantity of underlying data and low constraints in terms of processing time as experiments are usually carried out in a sequence of

batch jobs. Keyphrase extraction on the other hand is not (as much) concerned with conforming to a pre-defined vocabulary, and it is the mere descriptiveness of the output with respect to the topics apparent in the underlying document which defines the success of the approach.

While the output of ATE – word sequences referring to specialised nomenclature or terminology – certainly has descriptive character and therefore could be seen as what keyphrase extraction should provide, the opposite does not hold. Extracted keyphrases are not meant to conform to some controlled vocabulary (which would be *keyphrase assignment*), they are meant to give a brief, concise summary of the occurring topics in a text.

However, still a number of techniques in both fields are similar, and as a substantial amount of the considerations made in ATE could be useful for keyphrase extraction as well, I will survey the most relevant approaches of both fields subsequently, starting with the corpus oriented terminology extraction, before – finally – turning to single document oriented keyphrase extraction.

2.3.1 Corpus Oriented Terminology Extraction

With the advent of available large amounts of machine readable textual data, a number of approaches for ATE and collocation extraction have been proposed over the past 15 years, in a variety of settings.

Bootstrapping the creation of terminological structures from large, domain specific corpora has been one of the purposes, as exercised by Daille [17] and Feldman et al [24], whereas the continuous (semi)automatic update of existing terminological databases has been the focus of Bourigault [5] and Collier and colleagues [15].

Both settings are also closely related to the efforts undertaken by those parts of the ontology learning community concerned with the acquisition of ontological/conceptual structures from text, as witnessed by the recent workshops and activities in the context of OntoLex2000 [18], OntoLex2002 [19], OntoLex2004 [20], OntoLex2005 [21], OntoLex2006 [22], and OntoLex2007 [23].

[18] `http://www.ontotext.com/OntoLex/index2000.html` – retrieved 2008-08-07
[19] `http://www.ontotext.com/OntoLex/OntoLex02.html` – retrieved 2008-08-07
[20] `http://www.loa-cnr.it/ontolex2004.html` – retrieved 2008-08-07
[21] `http://www.ilc.cnr.it/ontolex2005/` – retrieved 2008-08-07
[22] `http://www.loa-cnr.it/ontolex2006` – retrieved 2008-08-07
[23] `http://olp.dfki.de/OntoLex07/` – retrieved 2008-08-07

Linguistic Approximation to Term Extraction: C-Value/NC-Value

Frantzi et al [29] present an approach to ATE, combining linguistic and statistical information. They assume the availability of shallow linguistic information (part-of-speech tags and a stopword marking), and experiment with different linguistic filters to extract candidates (mostly noun chunks) exhibiting specific category patterns, such as $NounNoun^+$, Adj^*Noun^+ and a more liberal filter accepting a greater variety of patterns.

Their algorithm, *C-value/NC-value*, is split into three stages, where (1.) *C-value* assigns an initial, linguistically and statistically inspired measure of *termhood* to candidates, (2.) a context weight is determined for content words (noun, verb and adjective) found at the boundary (directly before or after) a term, and (3.) *NC-value* revisits the *termhood* values obtained by (1.) and re-ranks the candidate list by a contextually enriched measure, as obtained by (2.), combined with C-value .

The three steps will be covered in more detail here:

1. C-value starts by extracting a set of term candidates from a corpus by the previously mentioned linguistic filter. Before a termhood score for a candidate a is computed, an important distinction – whether a appears as a nested term in other candidates (a is substring of another candidate) or not – is made, triggering a dedicated treatment of nested terms.

   In case of non-inclusion, the termhood value is simply the logarithm with base 2 of the length in terms of words of the candidate [24] $log_2(|a|)$, multiplied by the observed frequency of $f(a)$.

   However, in case a is a nested term, the authors reduce the importance of the frequency, as the distribution of a is wider spread among different (larger) candidates, denoted by the set T_a. The amount of the frequency reduction equals the sum of the observations of every element in T_a, scaled by the cardinality of T_a:

$$C\text{-}value(a) = \begin{cases} log_2(|a|) \cdot f(a) & a \text{ is not nested} \\ log_2(|a|) \cdot (f(a) - \frac{1}{P(T_a)} \cdot \sum_{b \in T_a} f(b)) & \text{otherwise} \end{cases}$$

2. To incorporate a weight reflecting contextual information, the authors assemble a list of important term context words (words directly preceding or succeeding a term), to have a mechanism at hand for promoting terms which additionally combine with lexical items that are important for the given domain. The list of

[24] slightly boosting terms with sequences of more than 2 words

context words is yielded by extracting the top 33% of the ranked list obtained by C-value in the previous step, considering the term candidates found there as true terms. For each term contained, nouns, verbs and adjectives in preceding and succeeding position are extracted from the corpus, as these are the morpho-syntactic categories carrying lexical meaning – however, the assumption here is that C-value already gives a satisfactory ranking in the top 33% of the resulting list. Now, for each context word candidate w, a context weight $weight(w)$ is computed, by calculating the relative frequency of the number of terms the context word combines with $t(w)$, still in the context of the top 33% of the ranking provided by C-value:

$$weight(w) = \frac{t(w)}{n}$$

3. With the previous two steps, the list yielded by C-value is revisited, re-ranking a term a by incorporating contextual knowledge, considering all context terms $b \in C_a$ it is appearing with:

$$NC\text{-}value(a) = 0.8 \cdot C\text{-}value(a) + 0.2 \cdot \sum_{b \in C_a} f_a(b) \cdot weight(b)$$

C-value/NC-value has been evaluated and applied in different languages and a number of different domains, most prominently in the biomedical domain [28] using English datasets and in a Japanese AI setting [1].

Extending C-Value/NC-Value: SNC-Value

Building on the foundations of C-value/NC-value, Maynard & Ananiadou propose an extra layer on top, the so-called SNC-value, incorporating syntactic cues, further contextual information and domain specific semantic knowledge [50, 49].

To achieve this, an additional measure, *Information Weight IW* is introduced, comprising of (i) syntactic knowledge (boundary probabilities, boundary designators), (ii) terminological knowledge yielded by the NC-Value approach, and (iii) semantic knowledge provided by the Unified Medical Language System (UMLS) [25] .

Syntactic knowledge is encoded in terms of (so-called) boundary words, or designators, by examining the syntactic category of words occurring directly before and after a known term or its context words (c.f. C-value/NC-value above). These observations are captured in a statistical model, and based on this, different weights are assigned to

[25] `http://www.nlm.nih.gov/pubs/factsheets/umls.html` – 2008-08-19

different syntactic categories, intuitively expressing that, for instance, a verb with an assigned weight 1.2 is a better designator for a true term than an adjective with weight 0.7 would be. Here, the syntactic weights *syn* determined for the categories, given as $\langle category, weight \rangle$ are $\{\langle Verb, 1.2\rangle, \langle Prep, 1.1\rangle, \langle Noun, 0.9\rangle, \langle Adj, 0.7\rangle\}$.

Terminological knowledge is derived from the output of the NC-value approach, similar to step 2 in the C-value/NC-value procedure, but this time focussing on terms instead of words, such that the weight CT obtained here is defined as sum of all context terms T_a appearing at the boundaries of each candidate term a:

$$CT(a) = \sum_{d \in T_a} f_a(d)$$

Again, it shall be noted that the top-33% of NC-value term candidates are treated as true terms here.

Semantic knowledge relies on structures provided by the UMLS metathesaurus [26] and semantic network [27]. Here, it is calculated to determine the semantic proximity between a candidate term and its context terms. Intuitively, a closely related context term should be more significant to a candidate term than an unrelated context term. If for a context term and a candidate term concepts in the hierarchy have been identified, two measures become relevant to compute the semantic similarity *sim*: the vertical position and horizontal distance, defined here as *positional weight pos* and *commonality weight com*, respectively, where *pos* is given as the combined depth in the hierarchy tree, and *com* is defined as the number of shared common ancestors multiplied by the number of words. The semantic similarity *sim* is given as

$$sim(w_1 \dots w_n) = \frac{com(w_1 \dots w_n)}{pos(w_1 \dots w_n)}$$

where $w_1 \dots w_n$ refers to the concepts identified in the knowledge structure, which correspond to the terms for which *sim* should be computed.

Now, armed with these measures, the information weight $IW(a)$ combining contextual syntactic, terminological, and semantic insights for a term can be defined as

$$IW(a) = \sum_{b \in C_a} syn_a(b) + \sum_{d \in T_a} f_a(d) \cdot sim_a(d)$$

[26] `http://www.nlm.nih.gov/pubs/factsheets/umlsmeta.html` – retrieved 2008-08-19
[27] `http://www.nlm.nih.gov/pubs/factsheets/umlssemn.html` – retrieved 2008-08-19

Similar to what NC-value does to an output yielded by C-value (re-ranking), SNC-value merely re-ranks the output obtained by NC-value, such that

$$SNC\text{-}value(a) = NC\text{-}value(a) + IW(a)$$

thereby enriching NC-Value with information calculated as importance weight IW.

SNC-value has been evaluated on a corpus of 800,000 eye-pathology records, and a direct comparison with the NC-value method confirms that the incorporation of the additional knowledge sources mostly yields an improvement in the ranking.

Exploiting Modifiability Characteristics of Terms

In a series of corpus linguistic experiments, Wermter & Hahn explore algorithms exploiting the so-called *limited* and *paradigmatic modifiability* of collocations and domain-specific terms [76, 77, 78].

In [76] the authors argue that despite the repeatedly proven suitability of statistical methods for identifying collocations (log-likelihood, t-test, etc.), there has been a lack of acknowledging and incorporating underlying linguistic assumptions into algorithms for collocation testing. The assumption here is that (parts of) collocations are not substitutable (as, for instance, in 'to spill gut', 'gut' cannot be replaced by 'intestine'), or only to a limited extent modifiable (e.g., in 'to kick the bucket', 'bucket' cannot be modified by 'green', or 'plastic', as in 'to kick the plastic bucket'), without losing the meaning of the collocation.

To demonstrate their hypothesis that a linguistic oriented measure for collocation testing should perform well among the state-of-the-art statistical methods, Wermter & Hahn consider collocations adhering to the surface pattern *verb-prepositional phrase*, extracted from a 114-million-word German language newspaper corpus which had undergone part-of-speech tagging, sentence splitting and NP/PP-chunking. The extracted verb-prepositional phrases were represented as <P,N,V>-triples, where P corresponds to the preposition, N to the headnoun and V to the verb contained in each extracted construction. Additional lexical tokens such as determiners, adjectives, numerals, etc. associated with each triple were defined as *supplements*. Only <P,N,V>-triples which exceeded a frequency count $f \geq 10$ were taken into consideration, yielding 8,644 different candidate *types*, distributed over 279,350 candidate *tokens*. The candidate types were manually classified by three human judges as representing a collocation or not, which led to a set of true collocations comprising of 1180 triples, to be used as gold-standard.

To capture their linguistic assumption, that a true collocation should be associated with a very characteristic supplement pattern (ideally, the only supplement pattern), the authors define the modifiability measure $\mathcal{MOD}$ for a given <P,N,V>-triple PNV_{triple}, which is based on the relative frequency of the most often observed supplement, expressed as probability $\mathcal{P}(PNV_{triple}, Supp_k)$:

$$\mathcal{P}(PNV_{triple}, Supp_k) = \frac{f(PNV_{triple}, Supp_k)}{\sum_{i=1}^{n} f(PNV_{triple}, Supp_i)}$$

where f is the absolute frequency and n is the total number of distinct supplements. Intuitively, for a true collocation, one would expect very few supplements. Now, the modifiability for a given <P,N,V>-triple $\mathcal{MOD}$ is defined by the most likely supplement:

$$\mathcal{MOD}(PNV_{triple}) = argmax\mathcal{P}(PNV_{triple,Supp_k})$$

Thus, for <P,N,V>-triples where only few variations in the supplement can be observed, $\mathcal{MOD}$ is expected to be significantly higher than for <P,N,V>-triples with wildly varying supplement.

In order to justify the classification of a given <P,N,V>-triple as collocation, a significant amount of observations of the triple have to be made throughout the data collection, as currently the $\mathcal{MOD}$ measure is (so to speak) triple-internal, and does not reflect the frequency within the corpus (leaving the possibility open for it being a systematic mistake, specialised language, or a newly-coined term that has not yet found acceptance on a large scale, which would not justify the classification as collocation). To guard against this, a measure of collocativity $\mathcal{COLL}$ is being defined, based on the relative frequency of the <P,N,V>-triple in question in the corpus, again, expressed as probability:

$$\mathcal{P}(PNV_{triple}) = \frac{f(PNV_{triple})}{\sum_{j=1}^{t} f(PNV_{triple_j})}$$

where t corresponds to the total number of distinct triple-types, and

$$\mathcal{COLL}(PNV_{triple}) = \mathcal{MOD}(PNV_{triple}) \cdot \mathcal{P}(PNV_{triple})$$

Thus, $\mathcal{COLL}$ is expected to rank true collocations at the very top of the list of candidates, whereas non-collocations should obtain a very low $\mathcal{COLL}$-value, pushing them towards the bottom.

A sequence of evaluation experiments, where the new algorithm is compared against established statistical methods (t-test, log-likelihood) and pure frequency, confirms the

hypothesis, accomplishing significantly better results in terms of recall and precision in each single segment of the result set.

In a follow up study, operating on a 104-million word corpus of 513,000 English Medline [28] abstracts, Wermter & Hahn introduce the *P-Mod* measure [77] in order to identify biomedical terminology, exploiting the (so-called) *paradigmatic modifiability* of domain specific terms. After a shallow linguistic analysis including morphological normalisation (lemmatisation) of the nominal head, they consider noun-chunks of word lengths 2, 3 and 4, exceeding a cut-off frequency threshold of 10, 8 and 6, respectively, as candidate terms.

Their intuition – a true terminological sequence of words remains more 'frozen', or 'static' than a non-terminological one – is modelled by the binomial coefficient $\binom{n}{k}$, where n is the number of words in the sequence (i.e. the n of n-gram), and k is the number of slots which can be filled. For instance, a particular n-gram of size n=3 written as $\langle w_1, w_2, w_3 \rangle$ can have different slots k filled with k=1 $\langle k_1, w_2, w_3 \rangle, \langle w_1, k_2, w_3 \rangle, \langle w_1, w_2, k_3 \rangle$ and k=2 $\langle k_1, k_2, w_3 \rangle, \langle k_1, w_2, k_3 \rangle, \langle w_1, k_2, k_3 \rangle$, which are called selections *sel*. With k acting as a wildcard in any slot, the authors define a measure of limited modifiability *k-modifiability* mod_k for each number of k (the number of open slots) such that

$$mod_k(n\text{-}gram) = \prod_{i=1}^{s} \frac{f(n\text{-}gram)}{f(sel_i, n\text{-}gram)}$$

reflecting the probability of a modification taking place, which, based on the underlying intuition, should not happen in case of a true terminological sequence of words, resulting in a fairly high value of mod_k, whereas for wildly varying modifications in case of an open slot, mod_k is expected to be close to 0.

The *paradigmatic modifiability*, *P-Mod* for an n-gram is then given as the product of all its k-modifiabilities

$$P\text{-}Mod(n\text{-}gram) = \prod_{k=1}^{n} mod_k(n\text{-}gram)$$

resulting in a ranked list of candidates where n-grams exhibiting very limited modifiability are ranked at the top, whereas n-grams heavily varying in the combinations with other elements are ranked near the bottom.

[28] `http://www.nlm.nih.gov/pubs/factsheets/medline.html` – retrieved 2008-08-07

In an evaluation of *P-Mod* against UMLS, at the same time comparing their results with the commonly used ATE approaches C-Value and t-test, the authors report on a significant increase of performance in terms of precision and recall.

2.3.2 Document Oriented Keyphrase Extraction

Compared to ATE and collocation detection, keyphrase extraction has not received as much attention in the scientific literature, partly because ATE offers greater potential for continued research, partly because the output of keyphrase extraction has limited immediate scientific value, and as such, is treated more as an engineering exercise. Still, as motivated in chapter 1.1, the added value to the common user cannot be underestimated in an era of digital information overflow, where fast and simple methods to summarise textual content are needed urgently.

Taking noun phrase information into consideration, Barker & Cornacchian propose a simplistic keyphrase extraction implementation based merely on head-noun frequency and phrase length [3]. They conduct an evaluation relying only on human judges, however have to report 'spectacularly low kappa values' for inter-annotator agreement, before concluding that nevertheless their algorithm performs comparable with the Turney's *Extractor* algorithm [75] which is described in more detail below.

Matsuo & Ishizuka present an approach based on co-occurrence statistics of frequent words within the same sentences, using similarity and pairwise clustering methods [48]. They evaluate their algorithm in a user study on 20 technical documents where judges were asked to rate the top-15 predicted keyphrase candidates, with the possibility to supply a set of 5 indispensable keyphrases used for coverage calculation. The evaluation performance in terms of precision reached 51%, whereas coverage is reported at 62%.

Although mainly concerned with terminology extraction, Sclano & Velardi propose an algorithm performing well on single documents [67]. They introduce three different measures *domain pertinence*, *domain consensus* and *lexical cohesion*, which are incorporated into a single weight expressing the relevance of a term candidate. The use of such measures suggests the existence of a substantial knowledge base in the background, holding (lexical and statistical) information about different domains which is required for calculating the values proposed by mentioned measures. Unfortunately, it remains unclear how the authors determine the set of term candidates, a fact that is of great interest as a lot of noise is usually cancelled out by reducing the number of

word sequences considered as candidates. A website [29] provides additional information about the ongoing interactive large-scale evaluation of the approach, and the algorithm is exposed as a web application for demo and testing purposes.

In a (semantic) desktop scenario, Chirita et al [12] present a keyword extraction approach taking into consideration the information stored on a user's computer, with the aim of producing context-relevant predictions to the user when browsing web pages. They conduct an evaluation comparing different standard methods from various fields (frequency, log-likelihood, lexical compounds), reporting on relatively high precision scores between 70% and 80%. However, the basis for precision values are only the first n=4 tags returned by the system, and as only precision is reported, it leaves much room for speculation about the performance of the algorithm for generating all possible relevant tags, a measure usually computed by *recall*. Although the authors give some details according to implementation, it appears there is no freely available software component providing the functionality that has been reported on.

Next, two important machine learning oriented approaches to keyphrase extraction (*GenEx* and *KEA*) will be presented in a more detailed manner. It is interesting to note that, unlike the ATE efforts outlined in the previous subsection, neither of them makes use of programmatic shallow linguistic processing such as part-of-speech tagging or chunking.

Keyphrase Extraction as Learning Problem Using a Genetic Algorithm

Turney presents the keyphrase extraction problem from a machine learning perspective [75], reporting on the performance of two different learning algorithms.

Keyphrase extraction here is encoded as a supervised classification problem, where the task is to predict whether a given sequence of words is a positive or a negative example for a keyphrase. The two machine learning algorithms are the more general *C4.5* decision tree classifier, and *GenEx*, a symbiosis of the genetic algorithm *Genitor* and *Extractor*, a parametrised extraction algorithm encoding specialized procedural domain knowledge.

A series of experiments is conducted on five small-sized corpora, each corpus consisting of documents varying in length [30]. To demonstrate the general purpose nature of both classifiers, the document collections were assembled from different domains, and only documents were taken into consideration that had author-assigned keyphrases attached. Along those lines, the five compiled corpora consisted of (i) 75 Journal articles

[29] `http://lcl2.uniroma1.it/termextractor/` – retrieved 2008-08-02

[30] Document length is defined in terms of words throughout this thesis.

(from 5 different journals) (ii) 311 Email (institute internal) messages, (iii) 90 web pages indexed by Aliweb [31], (iv) 141 web pages taken from NASA Research [32], and (v) 35 documents taken from the FIPS [33] web page . For the two learning algorithms, $\frac{3}{4}$ of the email and journal corpora were used as separate training data, with the purpose of creating different classifiers, generalising to shorter documents (trained by the email subset) and longer ones (trained by the journal subset).

Turney's approach operates internally on stemmed phrases using the Lovins stemmer [44] for *C4.5* and stemming by truncation for GenEx, both more aggressive stemming techniques than the well-known Porter stemmer [57]. A phrase here is treated as an n-gram of varying size between 1 and 3 words, that does not contain any stopwords. In both learning tasks (*C4.5* and *GenEx*), 12 phrase parameters were identified, such as the number of words per phrase, the first occurrence of a phrase in the given document, the frequency of a phrase in an underlying document, etc., encoding the features of a given phrase.

The *C4.5* algorithm was trained on 9 of the 12 parameters, ignoring 2 features and using 1 feature as class prediction value. The input for the decision tree training was the whole set of possible phrases (stemmed n-grams not containing stopwords) of each training set, resulting in a very large proportion of feature vectors used as negative examples, compared to the very small amount of feature vectors used as positive examples, a problem that gave rise to the *GenEx* approach, a combination of two algorithms, *Genitor* and *Extractor*.

To reduce the amount of negative training examples, Turney integrated so-called procedural domain knowledge captured by the *Extractor* algorithm with *Genitor*, a genetic *steady-state* algorithm. *Extractor* is a parametrised 10-step extraction algorithm for stemmed keyphrases from a document, and its parameters are essentially the same 12 as the ones previously used by the *C4.5* approach. The procedure pursued by *Extractor* is a complex orchestration of steps, ranging from noise filtering, stemming words by truncation, counting frequency of stems and observing their first occurrence,

[31] `http://www.aliweb.com/` – retrieved 2008-08-04 — Aliweb was one of the first internet search engines, and it was possible to specify a URL accompanied by keyphrases which were stored in the Aliweb index. Unfortunately, most of the Aliweb index is out-of-date and many URLs ceased to exist.

[32] the originally given URL `http://tag-www.larc.nasa.gov/tops/tops_text.html` has ceased to exist

[33] `http://www.itl.nist.gov/div897/pubs/` – retrieved 2008-08-04 — The US Government Federal Information Processing Standards (FIPS) comprise a number of resources documenting various guidelines to be followed by government institutions when dealing in the fields of security, authentication and interoperability.

ranking stems by frequency, selecting top-n stems to be used as cue-stems for considering a reduced set of phrases (n-grams), computing scores for considered phrases, filtering phrases after linguistic criteria, eventually resulting in a greatly discriminated set of candidates predicted as keyphrases. Most of the steps in this procedure are governed by the parameters given at start time of the algorithm (e.g., truncation-length, various thresholds and boosting factors, etc.), and it is the task of the genetic algorithm *Genitor* to find an optimal population of parameters for the Extractor algorithm, fine-tuning each feature to maximise its fit on the training data. Throughout the *Extractor* algorithm, only very weak and simplistic linguistic assumptions are made, as it appears regular expressions on word endings are used to detect adjectives (for instance, words with suffixes '-al', '-ic', '-ible'), and a list of common verbs is used to distinguish nouns from verbs.

Turney reports on evaluation results where the *GenEx* approach consistently achieves better results than the best performing setup of the *C4.5* algorithm when applied on the test set of the 5 corpora, raising precision between 2% and 10%, depending on the corpus. However, the overall average precision in this experiment varies between 11.8% and 29% (with average mean at 19.58% and median at 19.15%). In an anonymous, web-based user evaluation of the *GenEx* approach over a seven month time period, annotators were asked to provide web pages and gauge the quality of the top-7 predicted keyphrases, ultimately achieving a 62% rate of predictions rated as good.

Unfortunately, the referenced web site containing a demo version and further information [34] was defunct and could not be retrieved at the time of writing [35].

Keyphrase Extraction by Naive Bayes Classification: KEA

In another machine learning approach, Frank et al define keyphrase extraction as a naive Bayes classification problem [27]. The classifier is trained on a document collection with author assigned keyphrases which are considered positive example instances, and after training is expected to be able to predict the probability of a candidate phrase being a keyphrase.

Phrases in this context are essentially n-grams of up to size n=3, and they possess a binary feature – the so-called *a priori* probability – stating whether they should be considered as keyphrase candidates or not, which helps to greatly discriminate the number of phrases considered, reducing the vast amount of negative examples during

[34] `http://extractor.iit.nrc.ca/` – unretrievable
[35] August 4^{th}, 2008

training and testing, a problem also encountered in the C4.5 application of Turney.

The *a priory* probability for each phrase from the list of all phrases is determined by (i) eliminating those phrases (n-grams) that have a stopword prefix or suffix, (ii) discarding phrases consisting only of a single proper noun, (iii) converting the phrases to lower case and performing Lovins stemming [44] on each word contained, and eventually (iv) eliminating those stemmed phrases occurring only once in the document [36]. The remaining list of phrases are assigned an *a priori* probability of 1, all excluded phrases obtain *a priori* probability of 0.

Now, to encode each phrase as feature values, only two properties (besides *a priori* probability) are used: (i) the TFIDF value of a phrase *phr* with respect to the underlying document *doc* and the training corpus (training collection) C given as

$$tfidf(phr, doc) = P(phr|doc) \cdot -log(P(phr|C))$$

and (ii) the relative position *dist* defined as the first occurrence of phrase *phr* starting at word w_k in document of length $|doc| = \{w_0, \ldots, w_n\}$ given as

$$dist = \frac{w_k}{|doc|}$$

These features were converted to discrete values with the purpose of yielding better results, as it was easier for the Bayes classifier to generalise over discrete nominal data than over real numbers. The probability of a phrase *phr* being classified as a keyphrase is thus given as

$$P(phr|tfidf \cap doc) = \frac{P(tfidf|phr) \cdot P(dist|phr) \cdot P(phr)}{P(tfidf \cap dist)}$$

where $tfidf$ and *dist* are the discrete values of the observations described above, and $P(phr)$ is the *a priori* probability of the phrase being a keyphrase candidate.

In a later revision, this model is extended to accommodate domain specific knowledge, by observing how often a specific phrase *phr* has appeared as author-assigned keyphrase in the training data C, again using discretized values:

$$P(phr|tfidf \cap doc \cap C) = \frac{P(C|phr) \cdot P(tfidf|phr) \cdot P(dist|phr) \cdot P(phr)}{P(tfidf \cap dist \cap C)}$$

However, the authors recognise that training on a specific domain requires substantial effort in terms training data size to yield significant improvements.

[36] so-called hapax legomena

In a direct comparison with the *GenEx* algorithm on largely the same document collections, and with output scaled to the same number of phrases as in Turney's experiment (output size=7 keyphrases), KEA and GenEx are found to behave equivalently in terms of the amount of author-assigned keyphrases which have been predicted correctly.

KEA has been applied in a number of settings, such as in interactive document summarisation [37] and keyphrase extraction in a digital library scenario [38]. In the latter experiment, which also includes a detailed qualitative and quantitative evaluation, the proximity in terms of performance of KEA and Turney's approach was confirmed again. For quantitative evaluation, matching KEA's output against author assigned keyphrases, depending on classifier model and output configuration, precision values between 6% and 28% are reported (with mean at 12.43% and median 11%), whereas recall values vary between 5% and 32% (with mean at 16.38% and median at 17%). Qualitative evaluation results, where human judges were asked to rate predictions make by KEA for a number of documents, vary between 50% and 64.6% for approved keyphrases predicted by the system.

The KEA package is available for download as free software [37], however the required training step, where at least 20 documents with assigned keywords have to be used as training collection in order to obtain useful results increases the burden for out-of-the-box application by the ordinary user.

2.4 Summary

At the beginning of this chapter, the relevance of keyphrase extraction for metadata creation from textual documents has been outlined, and the scenario of the semantic desktop has been introduced. Subsequently, NLP techniques crucial for statistical processing on small textual data have been summarised.

Keyphrase extraction has been characterised and distinguished from Automatic Term Extraction (ATE) and Information Extraction (IE), and the state-of-the-art has been reviewed. The discussed ATE-based approaches are not document-centric, and thus operate on large corpora to extract collocations and domain-specific terms. However, they do consider shallow linguistic information extensively, and their statistical algorithms are linguistically inspired.

[37] `http://www.nzdl.org/Kea/download.html` – retrieved 2008-08-04

Both reviewed approaches to keyphrase extraction utilise machine learning techniques, and only to a limited extent linguistic information, to achieve optimal performance, however only KEA is available as freely usable tool. Because of its machine learning nature it requires training before it can be used with reasonable results, which imposes a high burden for common computer users. Therefore, in order not to hamper adoption, one of the characteristics for the tool presented here must be its simple use, not requiring user action such as training on documents before it can be used.

The following two chapters constitute the main contribution of this thesis. Chapter 3 presents use-cases, underlying assumptions, design decisions and implementation of the various necessary linguistic preprocessing steps, before chapter 4 elaborates in detail on the mechanisms found in the keyphrase extraction algorithm.

Chapter 3

Design

This chapter will introduce a number of real-world scenarios which could benefit from an approach of content-based keyphrase extraction on the document level. After an analysis of those use cases, a number of desired properties for the tool are extracted. Based on these desiderata, the implementation of necessary preprocessing modules is discussed, as they provide some of the data consumed by the keyphrase extraction component, which is described later in chapter 4.

3.1 Use Cases on the Desktop

As introduced in chapter 1, the exponential increase of storage capacity due to affordable pricing is leading to an information overkill not only on the web, but also on the desktop. Textual documents, rather small in their nature (compared to digital music or video data), easily are lost and scattered across folders on the file system, with only the filename remaining as an anchor as a quick, first-stop description. Once the location of the file is forgotten, it is increasingly hard to retrieve at a later point in time. Early adopters of desktop search tools such as Google desktop [1], Strigi [2] or Beagle [3] are able to experience an improvement in the retrieval of certain documents storing textual data, however mentioned tools are limited to the task of searching and retrieving, while keyphrases are used to provide a condensed way of describing content, a so-called content footprint. Thus, the two techniques are applied in complementary scenarios, information retrieval and content description.

Retrieval is commonly achieved via an inverted full text index over a document

[1] `http://desktop.google.com/` – retrieved 2008-07-29
[2] `http://strigi.sourceforge.net/` – retrieved 2008-07-29
[3] `http://www.beagle-project.org/` – retrieved 2008-07-29

collection (corpus), and the significance of the contained documents with regard to some query is established by measuring their similarity (TFIDF). While structures underlying search can do little in terms of describing content, the structures underlying description could greatly enhance search capabilities: For humans, it is not feasible to browse over the (inverted) full-text index trying to gain some understanding about some document – however providing a view on a list of given keyphrases is easy to implement (given the restrictions of screen size, etc.) and will help the user significantly to gain a first understanding about the content. Adding keyphrases as index terms to the metadata description (of a document) would allow search engines to harvest precisely this metadata, adding a layer of exact content description to their index, which could claim priority during ranking of the resultset when a search is performed.

As described in chapter 2.1.2, with the advent of semantic web technologies on the desktop, the so-called semantic desktop, lifted and formalised content metadata pave the way for a number of interesting application scenarios, enabling an increase in user experience, ranging from simple **semi-automatic meaningful indexing** over **visualisation** to new ways of **recommendation** and **collaboration**.

3.1.1 Content-based Indexing & Tagging Recommendation

With the rise of the Web2.0 movement, tagging [4] has become the quasi-standard for categorising content for internet media such as blogs and web-pages. Tags are manually assigned by the author, and increasingly used on desktop systems, to describe a diversity of desktop items such as textual files, digital music, and so on.

Recently, with the 4.0 release of the popular Linux KDE system, it is possible to assign tags (and ratings) to any desktop resource. Tag metadata are modelled in RDF and persist in a global datastore, where applications can access them in order to produce generic views with associated metadata, so-called (s)mashups such as Konduit [52]. For example, a scientific article carries metadata about its author, who also happens to have an entry in the user's address book and has been tagged as appearing on one of the user's photos. As address book entries and photo tags are also persistent in the metadata store, a third application could provide a gallery view of all collaborators the user *knows* [5].

[4] loosely associated terms (free form descriptions of content)

[5] where *knows* is interpreted implicitly by the fact that persons know each other if they have established a form of correspondence such as per email.

Semantic Notetaking Tool: Semn

The semantic note taking tool Semn [6] builds on the semantic desktop release Nepomuk-KDE [7] for KDE4 [8], and has been designed to accommodate a number of brief, personal interlinked records. With access to the metadata store, it is capable of consuming and interlinking available information, as well as contributing to the metadata store by creating its own metadata. The nature of the tool – handling textual information – and the need to briefly annotate the data with a small number of descriptive terms suggested the use of the keyphrase extraction application. The amount of information stored on each individual note is rather small, thereby creating a particular challenge for the keyphrase extraction component. When the user requests the generation of keyphrases for a particular note, she is presented with a list of relevant terms describing the content, and it is up to her which suggestions to accept. Accepted keyphrases are stored as tags in the system wide metadata store and can be used to generate views of notes which have been associated with a given term, very much in the fashion of the well-known blog interfaces, as seen in figure 3.1.

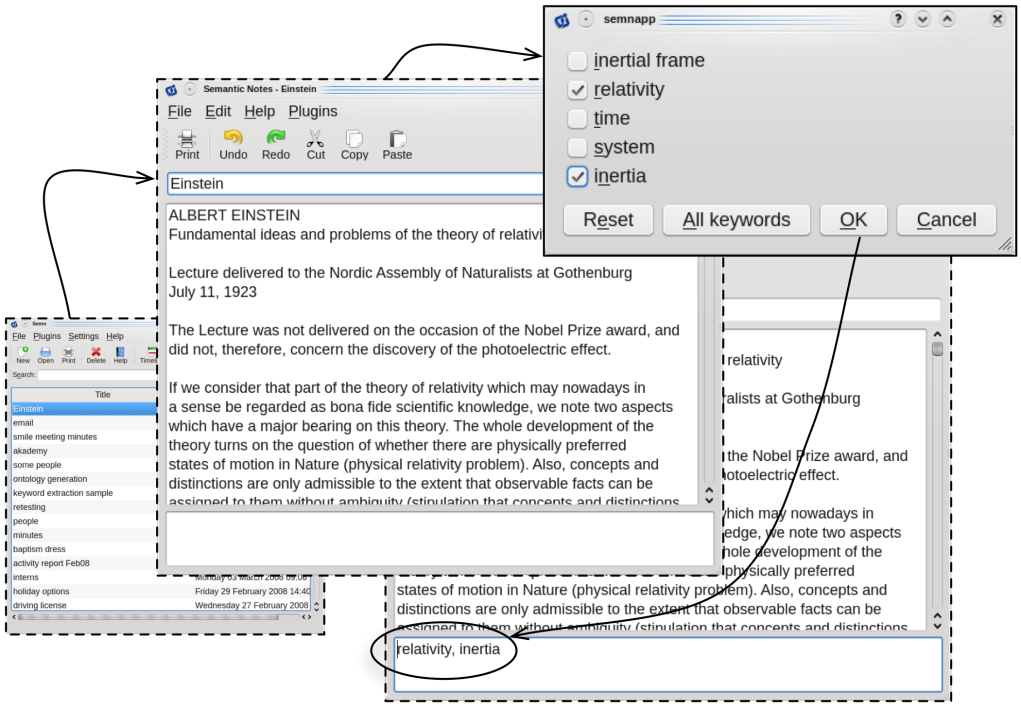

Figure 3.1: Semantic Notetaking Tool Screenshot

[6] `http://smile.deri.ie/projects/semn` – retrieved 2008-08-19
[7] `http://nepomuk.kde.org/` – retrieved 2008-07-29
[8] `http://www.kde.org/` – retrieved 2008-07-29

Here, the keyphrase extraction tool acts as a middleware service on the operation system, communicating with Semn via DBus [9], a popular message-bus for Linux, providing an implementation of an interprocess protocol (IPC). In theory, this approach enables other applications and desktop resources on Nepomuk-KDE to easily consume the keyphrase extraction service.

sClippy

As discussed in chapter 2.1, it is of great interest to transform implicit content-related metadata to its explicit state, lifting it to expressed, embedded metadata as a first step. Achieving this will result in a number of advantages, most importantly can the metadata be simply looked up if desired, leaving it much easier for metadata harvesters to make the data accessible in the systemwide RDF store. The desktop tool sClippy is one such lifting application, utilising the keyphrase extraction tool under the hood. Desktop files such as PDF documents are simply dragged and dropped into the application, and the user may start a batch job which analyses each document and generates metadata suggestions, which need to be confirmed by the user before they are permanently added to the document as embedded metadata.

While the scenarios described here predominantly deal with lifting keyphrases as descriptive content-metadata from their implicit state (within documents) to an explicit form, storing them either in the respective metadata fields of the document structure or the metadata repository of the desktop, the following scenarios build up on that in such a way that they make use of the process that has been described along these lines.

3.1.2 IVEA: Information Visualisation

The information visualisation tool for exploratory document collection analysis (IVEA) is a desktop application aiming to aid the user in the process of information gathering on a corpus of textual documents [72]. It adopts a user-centric stance by considering the person-specific interests, which also may change over time. Along those lines, IVEA operates under the assumption that a PIMO ontology (cf. section 2.1.2) exists, specifying the predominant concepts relevant for a given user. Those concepts are modelled as classes in the PIMO ontology. Class instances are used as queries against the corpus, the result of such queries is displayed as visual distribution over matching documents. Finer grained information, such as the distribution of

[9] `http://www.freedesktop.org/wiki/Software/dbus` – retrieved 2008-07-29

given instances within a given document is also available. Additional matrix-based displays representing instance views over documents can be generated, supporting the user in his exploratory data analysis. Also, during the information gathering process, the PIMO ontology can be enriched with additional classes or instances, in the latter case grounding it to the textual content of the documents.

In a previous version, IVEA proposed nouns and noun chunks as candidates purely based on their frequency, to be considered for ontology enrichment, which generated a number of unwelcome predictions, such as highly ranked short generic terms. Recently, the keyphrase extraction approach described here has been integrated with IVEA, with the hope of improving the candidates proposed for extending the PIMO, and first experiments have been encouraging. Figure 3.2 displays a screenshot of a recent

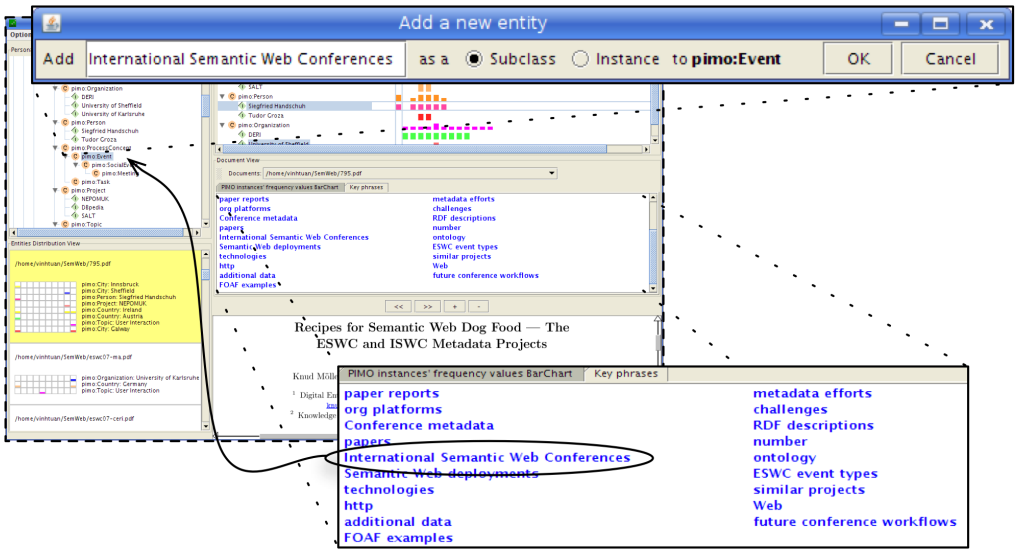

Figure 3.2: Information Visualisation Screenshot

version of the tool, giving an exemplary overview of the workflow when exploiting the keyphrase candidates for ontology enrichment. More information about IVEA is available on its dedicated web page [10].

3.1.3 Embedding into the Semantic Desktop

The keyphrase extraction has also been deployed as a service for NEPOMUK-Eclipse [11], a reference implementation of the social semantic desktop as proposed by

[10] `http://smile.deri.ie/projects/ivea` – retrieved 2008-08-19
[11] `http://nepomuk-eclipse.semanticdesktop.org/` – retrieved 2008-08-19

the NEPOMUK consortium [12]. Section 2.1.2 has already given a general introduction of NEPOMUK and its architecture, for more detailed information also see the NEPOMUK deliverables D6.1 [33] and D6.2.A [60].

The keyphrase extraction functionality is part of the TextAnalytics component, which offers a wider spectrum of NLP-based services, such as information extraction and speech act detection. The functionality can be accessed in a very simple way by any component residing on the desktop, for suggestion of free-form associated keyphrases for textual documents. The approach stands in contrast to a knowledge driven component for information extraction, which only recognises instances of classes that are already contained in the knowledge base, suggesting a use-case on the NEPOMUK desktop where the two approaches complement each other. Moreover, the output of the keyphrase extraction could be used as input for automatic summarisation algorithms.

3.2 Design Decisions

Based on the discussion in the previous section, it is now possible to derive a number of desired properties to be considered when implementing the tool.

3.2.1 Desiderata

Multilinguality

Although the English language is widespread and commonly used in academia and information technology, it is not exclusively apparent on the desktop. Localisation efforts in software give evidence that there is a need to provide services also to languages different than English, and academia should not have the luxury to choose which path to follow for convenience reasons. To address this issue, the approach undertaken here aims at providing a keyphrase extraction service to an audience not only restricted by the English language. However, it is increasingly hard to find free linguistic resources that are applicable off the shelf to different languages [13]. In the case of statistical or machine learning algorithms, it is often the case that models need to be trained for unsupported languages which means a considerable amount of know-how and annotated data is required.

[12] `http://nepomuk.semanticdesktop.org/` – retrieved 2008-08-19

[13] given the observation that for non-NLP persons it is already difficult to find (and utilise) free linguistic resources in their language of choice

From a cost-benefit perspective, for any given NLP component it seems utopical to achieve true multilinguality, however it can be approximated by supporting as many languages as possible at a time, and by designing the component in a clever way so it is easy to add further support for languages later on.

Single Documents, Multiple Formats & Cross Domain

The desktop scenario is radically different from most approaches in automatic term extaction which operate on large document collections, and often are restricted to a particular domain. Here, all sorts of single documents in the common file formats receive the focus of attention, meaning a significant reduction of data and a dramatic increase of noise. Input data is not cleanly prepared as are the large corpora available for conducting experiments in term extraction. The approach proposed here should be working on an everyday basis, on a largely unrestricted set of input data in the open domain. Therefore, it is expected to put significant effort into (i) filtering out noise at various stages of the processing, and (ii) to recognise circumstances where the standard way of treating input data is inappropriate (for instance, when data is too sparse to employ statistical algorithms) and an alternative path has to be chosen.

Robust, Forgiving & Cooperative

NLP components have successfully obtained the stigma of being slow and unreliable. While speed often is an issue, in particular when it comes to processing very large amounts of data, it can be an acceptable compromise if – at the end of the day – a superior result can be achieved. However, language data is highly irregular, and NLP components are notoriously known to exhibit non-deterministic behaviour at times when unexpected data sneaks in through the backdoor or the surrounding conditions change to an undefined state. Here, it is necessary to guard against a number of possible problems – for instance a linguistic resource could not be available for the language of the document. In such a case, either a controlled way of notifying the user needs to take place (graceful exit), or an alternative solution has to be found, with a possible decline in quality, but at least not leaving the user stranded.

3.2.2 Why Use GATE?

A framework making it easy to satisfy most of the previously discussed desiderata is the General Architecture for Text Engineering, or GATE [14] [16]. Besides its

[14] `http://gate.ac.uk` – retrieved 2008-08-19

system independent implementation, it provides document readers for the most popular file formats acting as container of textual content on the desktop, such as PDF, Microsoft Word, plain text, and so on, which is a huge advantage as it makes the utilisation of a dedicated text extraction component unnecessary. Moreover, a number of linguistic resources are readily available (sentence splitting, English part-of-speech tagger, morphological analyser, etc.), which significantly lowers the burden of starting out. Additional functionality can be implemented in a plugin-based fashion, which is important for distributing complex tasks into several units (plugins), each responsible for the solution of an isolated problem. The plugins can be loosely coupled into a processing pipeline in order to accomplish the original complex task, thereby relying and accessing persistent structures provided by the framework.

In the game for more than ten years, GATE has enjoyed a large user base, and it has been adopted by many industrial and academic projects, as a collection of GATE-related news headlines documents [15]. As a last, but nevertheless important point, GATE's liberal license scheme via the GNU Library General Public License [16] makes it possible to be deployed in a variety of projects without forcing hard decisions on the licensing scheme upon the actual project itself.

3.2.3 Addressing Multilinguality

Language Identification

As pointed out in chapter 2.2.1, in a pipeline of multilingual linguistic processing resources varying in complexity, language identification must be performed at the very beginning, as subsequent steps need to reflect the proper handling of the detected language resource. While there is not so much difference within Western European languages concerning less complex linguistic operations such as tokenisation and sentence splitting, with increasing complexity, appropriate decisions need to be made, as for instance substituting English Part-of-Speech Tagger with the French or German one, or loading respective grammars and dictionaries. Hence, for the reasons mentioned, it is obvious that a language identification has to be the first step done before any other linguistic components are applied.

[15] `http://gate.ac.uk/projects.html`, `http://gate.ac.uk/news.html` – retrieved 2008-08-19
[16] `http://www.gnu.org/licenses/old-licenses/library.html` – retrieved 2008-07-12

Part-of-Speech Tag Mapping

As introduced in section 2.2, part-of-speech tags are annotations at token level, assigning morphosyntactic categories to given words, such as *determiner*, *noun*, *verb*, *adjective*, etc.

Problematic when applied in a multilingual environment is the fact that different, non-overlapping tagsets have been established for different languages. The tagset mapping problem has been recognised elsewhere as witnessed by a number of contributions by the EAGLES [17] initiative, in particular the recommendations for the morphosyntactic annotation of corpora [18] and in the literature such as [71], and in particular [2].

Thus, in order to conveniently treat/access this data at a later stage, a mapping to a common, cross-language vocabulary has to be performed.

Multilingual Frequency Lists for Statistical Relevance Criteria

Section 2.2.2 introduced the notion of statistical NLP by computing association measures. In many corpus similarity and relevance ranking experiments, the British National Corpus has been chosen as a reference corpus [19]. Obviously, for such an approach to succeed in a multilingual setting, frequency lists for more languages are needed.

3.3 Making Use of the Framework

The GATE framework is pluggable, which means it offers core services and a unified data model, whereas most of the linguistic functionality is provided via plugins which embed their output into the structures provided by the framework. GATE plugins can be implemented as resources of 3 different sorts: (i) Language Resources (LR), (ii) Processing Resources (PR) and (iii) Visual Resources (VR).

3.3.1 GATE Data Structures

In GATE, a Language Resource has the sole purpose of storing and providing access to NLP relevant data (e.g., corpora, documents, lexicons or ontologies), whereas a Processing Resource typically performs data manipulation, using LRs as source (and

[17] Expert Advisory Group on Language Engineering Standards

[18] `http://www.ilc.cnr.it/EAGLES96/annotate/annotate.html` – retrieved 2008-07-15

[19] or more precisely, the lemma frequency lists compiled by Adam Kilgarriff, also see `http://www.kilgarriff.co.uk/bnc-readme.html` – retrieved 2008-10-22

possibly, as target or sink). Visual Resources provide a view for the graphical user interface, and shall not be treated in detail here.

In order to support the developer with a scaffold, GATE provides the so-called *bootstrap wizard*. A screenshot of the bootstrap wizard dialogue is presented in figure 3.3. After specifying the required fields and confirming the dialogue, a fully functional but

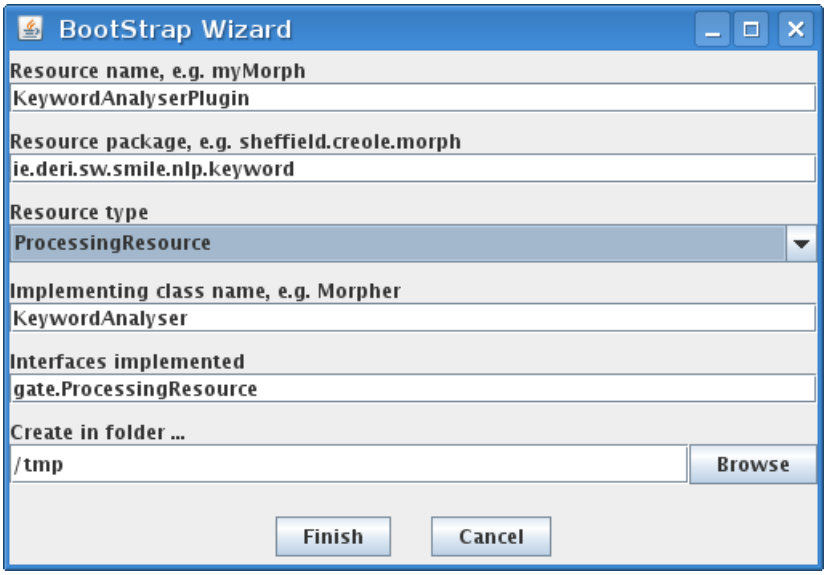

Figure 3.3: Bootstrap Wizard Dialogue

empty plugin skeleton is created from where rapid development of the plugin can start. The resulting structure yields the main class for the plugin and the so-called CREOLE XML file [20], which can be described as an access layer for resources shared between framework and plugin. The `creole.xml` also specifies the init-time and runtime parameters of the plugin, for which accessor methods have to be implemented, enabling it to be treated very much like a Java Bean [21] by the GATE framework. Furthermore, `lib` and `resources` folders exist, where 3rd party libraries and resources can be included and bundled together with the plugin, a feature that will be used for a number of plugins as described in the remainder. The main Java class of the GATE PR plugin has to implement two methods, namely `init()` and `execute()`, and additionally specify so-called *init-time parameters* and *run-time parameters*, which reflect the specification given in the CREOLE file of the plugin.

[20] an acronym for *Collection of REusable Objects for Language Engineering*

[21] `http://java.sun.com/javase/technologies/desktop/javabeans/docs/spec.html` – retrieved 2008-07-15

3.3.2 Implementing Preprocessors

All components described here have been implemented over the course of this undertaking. They have been deployed as a Processing Resource plugin for GATE, and in the following more details about the underlying techniques/technologies are given.

Language Identifier

As discussed in chapter 2.2 and chapter 3.2, achieving multilinguality implies identifying the language of a given input text at the very beginning of the processing pipeline so that appropriate linguistic processing resources can be selected at later stages.

The open source [22] library ngramj [23], implemented in Java and available under Lesser GNU Public License [24] provides a mapping from input text (given as string) to a sorted list of language identifiers (for example, $\langle en, de, fr, pl, \ldots, zh \rangle$), where the returned list is given in descending order of confidence for the languages available. The library relies on character n-grams and a competitive scoring algorithm over readily available n-gram profiles for a large number of European languages, among others (please see Appendix A.1 for all out-of-the-box supported languages).

In order to make this functionality available in GATE, a plugin implementing a ProcessingResource in form of wrapper has been created around the ngramj library.

As the ngramj-approach only needs very few characters ($\theta(10^3)$) of input text to stabilize on a prediction and produce a ranking of best-matching languages, the LanguageIdentifier drives efficiency even further: To save runtime, only 1000 consecutive characters from the input text are selected from an arbitrary offset, preferably from a central section of the document. The top element from the resulting ranked list of language identifiers is stored in the document model as `language` feature for easy access of succeeding ProcessingResources consuming the `language` feature, such as the StopwordAnalyser, POSTagMapper (both described in the remainder) and KeywordAnalyser (chapter 4). It should also be noted that during this effort, two additional n-gram profiles have been built from internet corpora for Irish and Chinese.

Stopword Analyser

Stopwords are a closed class of words that mainly fulfil syntactic functions, and as such barely are carrying lexical meaning. They comprise determiners, pronouns,

[22] `http://www.opensource.org/` – retrieved 2008-07-14
[23] `http://ngramj.sourceforge.net/` – retrieved 2008-07-13
[24] `http://www.gnu.org/copyleft/lesser.html` – retrieved 2008-07-14

prepositions, auxiliary verbs etc. and make up a large amount of the most frequently occurring wordforms in a given text. Due to their limited distinctive nature, stopwords have mostly been disregarded in information retrieval, where it has been common practice to skip them, only indexing so-called *content words*, as Manning et al point out in [45](chapter 2.2.2). Recently however, the big players in the search engine market such as Google or Yahoo! changed their policy towards inclusion of stopwords. [25]

From a term extraction point of view, where terms to be extracted consist of multiword phrases and not only of single words, it is important to maintain stopwords and not disregard them. This is motivated by the fact that stopwords might be included in terms to be found of descriptive nature for a given document, as for instance "the Earl of Essex", and with losing information about all stopwords it would be impossible to retrieve the term as a whole. It is however very useful to mark up a term as stopword, as this information can be utilised at a later stage when counting the frequency of phrases: For instance, the use of determiners for a given phrase might vary, resulting in significant data-sparseness especially in short texts. In this regard, marking up stopwords and carrying them along, but disregarding them when they only introduce noise results in a way of abstraction which is desired for certain actions.

To achieve this, available multilingual stopword lists have been acquired [26]. The lists have been utilised by Jacques Savoy and colleagues for a number of experiments for CLEF [27] submissions [63, 64], and were constructed following the guidelines given in [26].

The stopword lists are stored in one file per language. During processing, the language feature of the document is read initially, and on this basis the appropriate stopword list is selected. A simple iteration over the text tokens combined with a string comparison yields an additional binary feature for each token marking whether it is a stopword or not. A specific performance feature of the StopwordAnalyser is the dynamic loading behaviour of the stopword lists: in case a document collection is processed as a corpus, for each new document processed only the previously used stopword list is kept in memory. As corpora in the context encountered here usually are mono-lingual (meaning the appropriate stopword list is loaded only once), the result is a disk access of $\theta(1)$ at best-case, and $\theta(n)$ worst case, where n is the number of documents processed in the corpus.

The information introduced here at token level is consumed and exploited by the

[25] see also closely related blog discussion about Google policy change regarding stopword inclusion at `http://www.seofaststart.com/blog/stop-words-are-dead` – retrieved 2008-10-14

[26] `http://members.unine.ch/jacques.savoy/clef/index.html` – retrieved 2008-10-14

[27] Cross Language Evaluation Forum, `http://www.clef-campaign.org/` – retrieved 2008-07-15

components FrequencyAnalyser and KeywordAnalyser. More information about the stopword lists used by the plugin is given in the appendix B.1.

POS-Tag Mapper

As pointed out in the previous chapter, part-of-speech tags are not uniformly assigned for multiple languages English, French and German: The Hepple tagger [35] [28] used for English texts is based on a slightly enriched version of the Penn Treebank tagset [61, 47], as also documented in the dedicated page of the GATE documentation [29]. For German and French texts, the Tree Tagger [66] [30] is employed, and for German it relies on the Stuttgart-Tübingen tagset [65], whilst the French tagger produces annotations only specified in its accompanying documentation [31]. Luckily, for

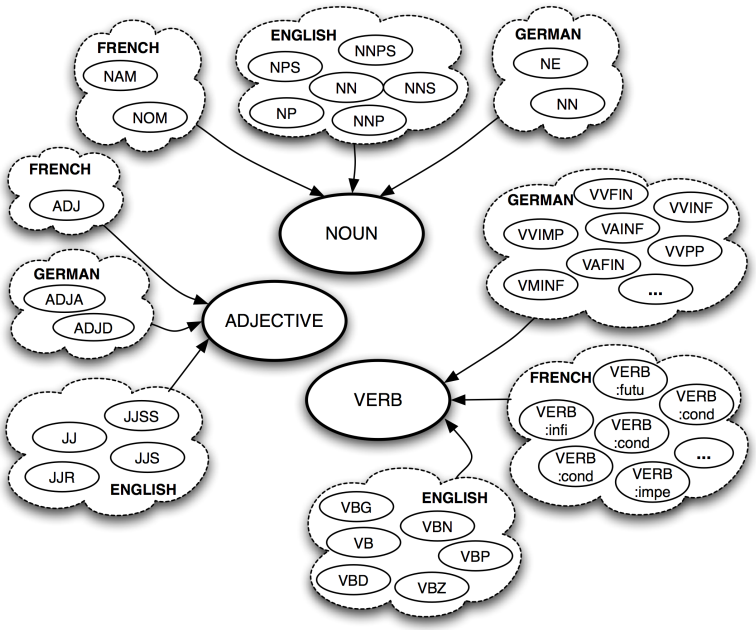

Figure 3.4: Mapping of language specific fine grained tags to coarse, unified categories

the approach described here, a fully fledged mapping of all possible tags is not necessary: it is sufficient to map the various fine grained content word tags (such as NN, NNP) for each language onto their respective coarse categories as sketched in figure 3.4.

[28] using a modified algorithm of the Brill tagger [8], such that learned transformation rules do not interact with each other

[29] `http://gate.ac.uk/sale/tao/splitap4.html` – retrieved 2008-07-16

[30] also see: `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/` – retrieved 2008-10-14

[31] `http://www.ims.uni-stuttgart.de/~schmid/french-tagset.html` – retrieved 2008-10-14

As the coarse grained categories comprise the cross-language elements [32] NOUN, VERB, VERBAUXMOD, ADJECTIVE, ADVERB and DETERMINER_OR_PRONOUN, the mapping to be performed is of a many-to-one nature. Instead of rewriting the original pos-tags, an approach of knowledge accumulation has been chosen – the mapping to the respective coarse category is simply added to the annotations at token level that are already present. Consumers of the here introduced coarse tags are FREQUENCYANALYSER and KEYWORDANALYSER, as outlined further below. The complete tagsets including their mappings for English, French and German are given in the appendix B.2.

English Noun Chunker

In draft 5963 on *Documentation – Methods for examining documents, determining their subjects, and selecting indexing terms* [33], the International Standardization Organisation (ISO) defines an indexing term as follows:

3.4 indexing term: The representation of a concept in the form of either

- a term derived from natural language, preferably a noun or noun phrase, or
- a classification symbol.

Here, the concept of a noun phrase has to be clarified, as from a linguistic point of view, noun phrases can be complex: they include a so-called *head-noun*, the semantic core which determines morphosyntactic features such as gender and agreement. The head noun may be further specified by *determiners* (a, the, these, etc.), *numerals* (one, two, ten, etc.), *quantifiers* (no, some, all, etc.), *premodifiers* and complex *post-modifiers* which can be of recursive structure and include phrases themselves as the following example illustrates:

	DET	PREMOD	HEADNOUN	POSTMOD
The doctor performed	*the*	*magnetic*	*induction tomography*	*quickly.*
	the	*magnetic*	*induction tomography*	*in the hospital.*

As in most of the cases it does not make sense to assign index terms that include determiners, numerals or quantifiers or even post-modifiers [34], it is assumed that the

[32] for the vast majority of languages

[33] `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=12158` – retrieved 2008-07-16

[34] unless the head-noun forms a semantic unit with the numeral/quantifier/post-modifier, as in *the Seven Dwarfs*, or *Lawrence of Arabia*, however such phenomena usually denote so-called (named) entities whose identification is not targeted by the work described here

term noun phrase as it is used in the ISO definition should be interpreted primarily as head-noun including its premodification. An investigation of articles from PubMed Central [35] that have assigned index terms confirms this assumption: from 20 randomly selected documents, of the 98 overall index terms, 97 were premodified head-nouns, whereas only 1 index term had the characteristics of a complex noun phrase (*An automated drug infusion system*) [36]. This insight is important from a computational perspective, as head-nouns including premodifiers can be yielded by so-called chunking techniques, which is significantly less complex than constructing noun phrases. [37] The GATE distribution is shipped with a plugin that performs noun chunking on English text, however, the resource has runtime performance issues. Therefore, an alternative GATE plugin yielding noun chunks has been implemented which runs significantly faster than the originally provided chunking plugin. The noun chunks generated by the original GATE component correspond almost in 100% of the cases to the noun chunks generated by the new plugin, which has been realized using a GATE finite state transducer by specifying a JAPE grammar with 11 macros and only 1 rule. The grammar introduces a new annotation type for noun chunks, whose instances are added to the default annotation set, effectively enriching the document with a higher-level structure between token and sentence annotations. The here implemented approach is a 1-to-1 adoption of the chunking strategy described in the LingPipe [38] part-of-speech tutorial, where a chunking implementation is covered briefly [39]. The only consumer of the annotations introduced at this stage is the KeywordAnalyser plugin. Please see appendix B.3 for the full specification of the JAPE grammar.

Frequency Analyser

Raw frequencies are the basis for any statistical calculation. In the context of statistical NLP, this implies counting lexical items such as word and lemma occurrences, which can be seen as a first approximation towards determining the most significant words in a given text.

However, raw frequencies have to be handled with care, as words in natural language exhibit a Zipf distribution [51] (cf. section 2.3).

[35] `http://www.pubmedcentral.nih.gov` – retrieved 2008-08-19

[36] `http://www.pubmedcentral.nih.gov/articlerender.fcgi?tool=pmcentrez&artid=1705490` – retrieved 2008-07-16

[37] A thorough discussion of the nature and shape of noun phrases is out of the scope of this thesis, however this example should be sufficient to motivate the use of chunking techniques.

[38] `http://alias-i.com/lingpipe/index.html` – retrieved 2008-07-16

[39] `http://alias-i.com/lingpipe/demos/tutorial/posTags/read-me.html` – the paragraph in question starts at $\frac{3}{4}$ of the webpage and is titles "Noun and Verb Chunking" – retrieved 2008-07-16

This is precisely the reason why here, exclusively (lemmas of) content words – and more specifically, nouns – are regarded as input for hypothesis testing, thereby disregarding problematic items at the top of the Zipf distribution, whereas rarely occurring lemmas/words with frequencies below 5 are also not chosen. Frequencies are counted for a number of different observations and stored as a document feature in a sorted HashMap (ordered by descending frequency), each representing a dedicated view over the word/lemma occurrences in the document: a) overall token frequency, b) overall lemma frequency, c) noun lemma frequency, d) verb lemma frequency, e) adjective lemma frequency .
Also, the document size in terms of word forms is counted and stored as a document feature as well as the lexicon size, which is the total number of different word types (or lemmas, cf. 2.2).

The sole consumer of the here generated frequency lists is again the KEYWORD-ANALYSER.

3.3.3 The Linguistic Food Chain: Who Consumes What?

This section gives a high level overview of the various components necessary for the proposed approach of keyphrase extraction.

As discussed before, it is inevitable to start the process with the **identification of the language** of any given document, in order to select subsequent processing resources appropriately. Following, the input text needs to be **tokenised** and **split into sentences**. This procedure is very much the standard in information retrieval. A **stopword analyser** marks up the tokens in the document structure with information whether they represent a stopword or not, rather than eliminating tokens found to be stopwords.

Arriving at a first junction now, it needs to be decided which **part-of-speech tagger** is used, depending on the information the language identifier has provided. In case of an English document, the *Brill tagger* shipped with GATE is utilised, in case of French and German the *Tree Tagger* is employed. This decision will also influence which component is used for **morphological analysis / lemmatisation**, as this functionality is also provided by the *Tree Tagger* in case of German and French texts. For English, the morphological analyser shipped with GATE as an additional plugin is used. Either way, tokens are now enriched with **part-of-speech** and **lemma** information. As the part-of-speech tags are heterogeneous and differ from language to language, a **mapping** plugin enriches the tokens with their coarse morphosyntactic

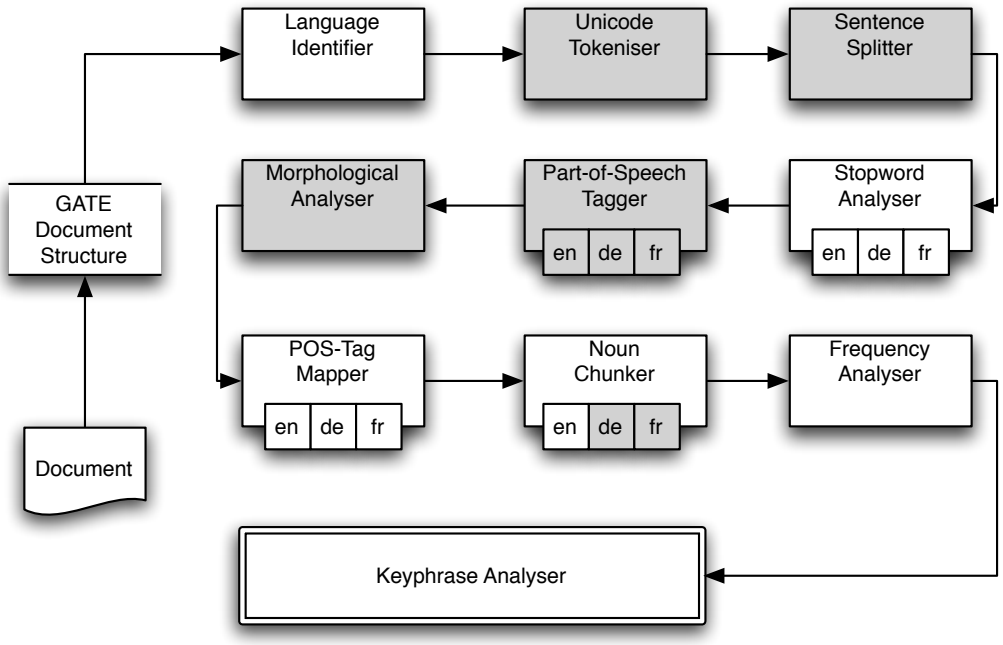

Figure 3.5: Plugin Architecture & Workflow

category, in order to be able to treat tokens in a unified way in the later stages of frequency and keyphrase analysis. Now, larger syntactic units are identified by the **noun chunker**, which again is dependent on the information provided by the language identifier, as different rule sets are loaded for English, German or French. Here, for German and French texts the MuNPEx chunker [40] is employed, whereas for English documents, a noun chunker implemented for this thesis is used. A **frequency analysis** step producing frequency lists of overall wordform and lemma occurrence is carried out. Besides the overall observations, frequency lists for all coarse grained morphosyntactic categories representing content words (nouns, verbs, adjectives) are also created, which is useful for convenient lookup during the lexical/statistical part of the **keyphrase analysis**.

The workflow as a whole is also depicted in figure 3.5, where the white boxes embody components or plugins that have (partly) been implemented as part of this thesis, whereas the grey boxes represent components that were available off the shelf as part of the GATE framework.

In the next chapter, the internal process of the keyphrase analysis is described in more detail.

[40] `http://www.semanticsoftware.info/munpex` – retrieved 2008-08-19

Chapter 4

Implementation

This chapter outlines in a detailed way the implementation of various steps undertaken to engineer a *knowledge-poor* keyphrase extraction mechanism suitable for single documents, possibly varying significantly in length, for a number of different languages.

To achieve reasonable results for a wide variety of possible input types (different document formats, very short/very long texts, use of different languages), the KeywordAnalyser plugin for GATE is consuming most of the aggregated information, represented as annotations in the GATE document structure, which were added by the previously described preprocessors. The component is divided into a number of subpackages, each of which are responsible for different aspects of the keyword analysis process. The distributed responsibilities include

1. the **(statistical) lexical analysis** to determine the most significant *single word terms*,

2. the **extraction of the previously identified single-word terms including their immediate contexts** to form *complex terms*,

3. the **grouping (or clustering) of similar complex terms**, and **selecting a representative** for each group (cluster) as a *keyphrase candidate*, and eventually

4. the **analysis of the extracted keyphrase candidates** in order to determine a *confidence score* in the context of the document in question.

Figure 4.1 offers a high-level overview of the first two parts of the keyword analysis (statistical computation over lexical items & extraction of complex terms), whereas the process of grouping (or clustering of) the complex terms and determining a candidate for each cluster is sketched in figure 4.2.

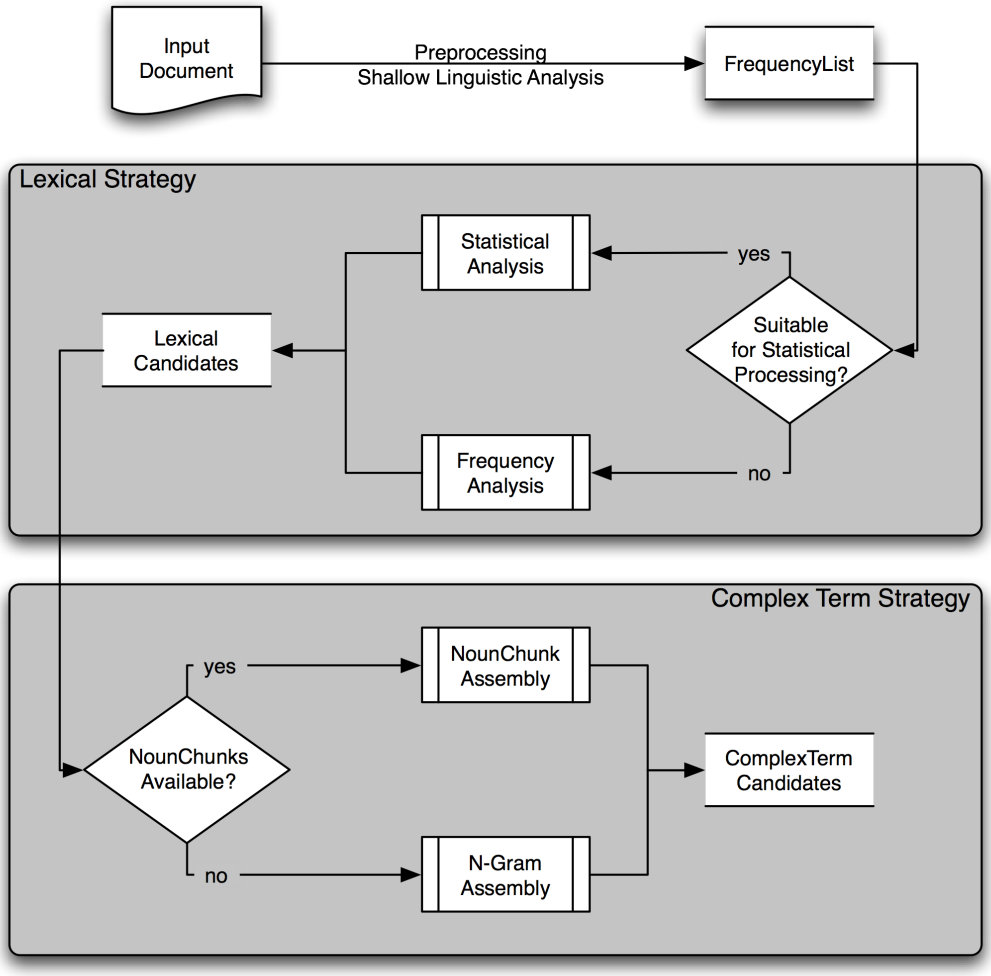

Figure 4.1: Keyword Analysis – Statistical Processing & Extraction of Complex Terms

4.1 Lexical Items & Important Words

Establishing a ranking of words/lemmas ordered by their significance for a given document can be achieved in multiple ways. The naïve way would be to sort the contained words by their number of occurrence, as done by the FrequencyAnalyser, and consider the top-n elements as important. As discussed in section 2.3, the most frequently occurring words in natural language are not at all significant for the content of a document, and more sophisticated methods exist, exploiting likelihood estimations and probability distributions. Unfortunately, statistical tests are not always applicable, and great care has to be taken to avoid statistical skewing, in particular if the data that is being tested (also called sample size) is sparse. As this tool should be used as a black box, deployed as a desktop component, it is of uttermost importance to make

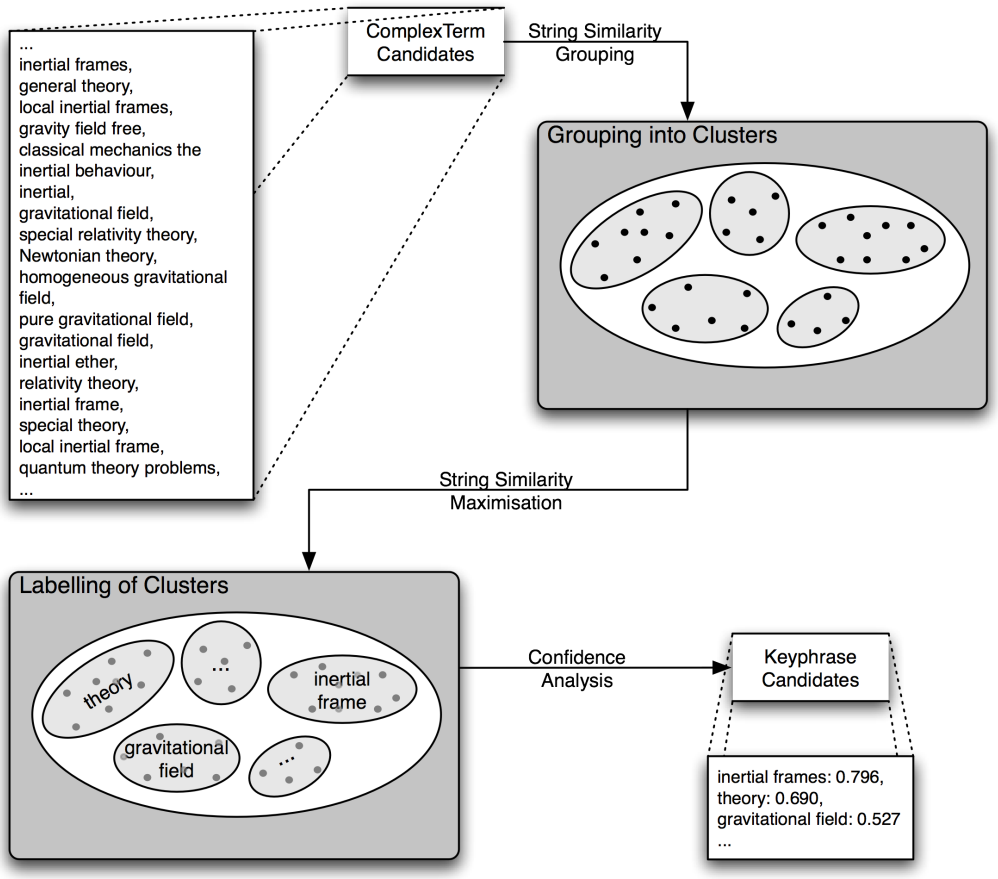

Figure 4.2: Keyword Analysis – Clustering, Labelling & Choosing a Candidate

the process robust and reliable for all possible variations of input.

The algorithm consumes the frequency lists for *noun lemmas* introduced by the FrequencyAnalyser.

The list *nounLemmaFrqList* is processed and its elements are aggregated into a unified list of candidate lemmas, called *candidateLemmaFrqList*, whereby the elements undergo a filtering procedure such that

(a) a selected element must have a frequency of 5 or more

(b) no more than the upper 25% or top 25 elements of the list (whichever comes first) are aggregated

This selection policy ensures that the aggregated list contains only elements which are suitable for statistical processing, up to a maximum of 25 elements. In the following, the statistical component is being described.

4.1.1 Statistical Processing

Here, the statistical component relies on *Pearson's chi-square* test [46](chapter 5), which belongs to the family of hypothesis tests. Other notable hypothesis tests are the *t-test* and likelihood ratios such as *log-likelihood* [23].

Hypothesis testing is frequently applied as standard measure when investigating phenomena observed in large corpora, and it has been found useful when establishing a significance ranking, e.g, for determining *co-occurrence* behaviour or identifying important words in a corpus. In essence, hypothesis testing formulates a so-called *null hypothesis* H_0, postulating that an event observed in a sample occurs more often than by chance, which in this context means to test whether a word that has an observed high frequency is really important: as shown in section 2.2.2, table 2.1, frequency alone is not always a reliable indicator of importance. Computing the statistic accounts for this, resulting in a ranking that, if sorted by the χ^2 just obtained, shall indicate the more important words at the top of the list (see also table 4.1 and 4.2).

Here, *Pearson's* χ^2 test is applied to assess the degree of corpus similarity when given two corpora, where the first corpus is the sample and the second corpus is a balanced reference corpus of significantly larger size.

To address concerns about the suitability of hypothesis testing for phenomena observed in natural language due to its non-random character — one of the core assumptions of hypothesis testing is a random distribution in the samples — raised by Kilgarriff [39] and recently re-iterated in [40], it shall be noted that here, the outcome of the test is not compared to a chi-squared distribution table, as the test is not used for maintaining or rejecting the null hypothesis. Instead, a list ranked by χ^2-value is produced which illustrates the difference in expected word frequencies between the sample and the reference corpus, as also pointed out by Kilgarriff and Rose [41]. Alternative statistical methods applied for similar purposes include *log-likelihood* [59], information theoretic measures such *pointwise mutual information* [1] [13] and *relative frequency* ratios as demonstrated by Damerau [18].

The underlying frequency lists for this calculation have been derived from a number of large, balanced corpora which have been compiled from the internet in context of the Web as Corpus initiative (WaCky) [2], and are available for various languages from Leeds University [3].

The χ^2 value for lemmas in *candidateLemmaFrqList* for sample *cSample* of size

[1] a variation is known as *pointwise Kullback-Leibler divergence*

[2] `http://wacky.sslmit.unibo.it/` – retrieved 2008-07-20

[3] `http://corpus.leeds.ac.uk/list.html` – retrieved 2008-07-19

	Corpus [a]	Medium-sized Sample [b]	
Lemma	Frequency	Frequency	χ^2
theory	27885	44	3278.51
field	40897	33	1226.86
law	80553	29	450.56
equation	3941	21	5362.19
system	134403	15	54.08
nature	30154	13	246.65
mechanic	2748	12	2509.15

[a] Large balanced internet corpus compiled by Leeds University; size 181,376,006 tokens; also see `http://corpus.leeds.ac.uk/list.html` – retrieved 2008-07-19

[b] `http://nobelprize.org/nobel_prizes/physics/laureates/1921/einstein-lecture.pdf` – retrieved 2008-07-19

Table 4.1: Lemma Frequency in Reference Corpus and medium Sample

N_1 and the reference corpus $cRef$ of size N_2 is calculated as follows:

1. treat $cSample$ and $cRef$ as random samples from the same population
2. for each lemma l in $candidateLemmaFrqList$ with frequency observation $o_{l,cSample}$ in $cSample$ and $o_{l,cRef}$ in $cRref$,
   (a) compute the expected values for both corpora:

$$e_{l,cSample} = \frac{N_{cSample} \cdot (o_{l,cSample} + o_{l,cRef})}{N_{cSample} + N_{cRef}}$$

$$e_{l,cRef} = \frac{N_{cRef} \cdot (o_{l,cRef} + o_{l,cSample})}{N_{cRef} + N_{cSample}}$$

   (b) summing up the squared error between observation $o_{l,cSample}$, $o_{l,cRef}$ and expectation $e_{l,cSample}$, $e_{l,cRef}$ (normalised by expectation) yields χ^2:

$$\chi^2 = \sum \frac{(o_{l,i} - e_{l,i})^2}{e_{l,i}}$$

Table 4.1 and 4.2 show how the ranking would adjust for the displayed lemmas of two documents if they were ordered by χ^2, promoting words/lemmas that are very specific to the respective document and penalising those which are used very commonly.

	Corpus [a]	Small-sized Sample [b]	
Lemma	Frequency	Frequency	χ^2
sector	19850	8	680.07
machine	20758	7	495.68
beam	27885	6	3022.14
particle	40897	6	1780.39
ring	30154	5	610.40
week	2748	5	75.91

[a] Large balanced internet corpus compiled by Leeds University; size 181,376,006 tokens; also see `http://corpus.leeds.ac.uk/list.html` – retrieved 2008-07-19

[b] `http://newsvote.bbc.co.uk/mpapps/pagetools/print/news.bbc.co.uk/2/hi/science/nature/7512586.stm` – retrieved 2008-07-19

Table 4.2: Lemma Frequency in Reference Corpus and small Sample

The lemmas obtained in this fashion are aggregated as so-called *CueTokens* in a *LexicalCandidateStore*, which is used later on to extract larger, more complex terms, such as chunks or n-grams. The scenario might arise that χ^2 cannot be computed because the lemma in question is not contained in the reference frequency list. In such a case, the lemma is assumed to be of some relevance if its frequency in the sample is above an empirically determined threshold of 5, and it is aggregated into the *LexicalCandidateStore* as well.

The reference frequency lists have been integrated into the plugin, and to save memory the same dynamic loading procedure is used as for the StopwordAnalyser, described in section 3.3.2. Appendix B.4 elaborates more on statistics about the lists and the corpora they were derived from.
Next, the method for the fallback procedure is sketched, which is employed in case frequency observations in the document are consistently low to guard against inappropriate use of the statistical method.

4.1.2 Guarding Against Sparse Data: Fallback to Frequency

If, as a result of a short input document (or data sparseness in general), the aggregated list *candidateLemmaFrqList* turns out to contain less than 10 elements [4], the statistical analysis is not performed. Instead, a new list of candidate lemmas is

[4] which means there were insufficient observations of lexical items with frequency 5 or more

constructed by repeating step (b) as described in section 4.1 – this time, no frequency filtering is applied as it is of importance to retain all possible content lemmas for frequency analysis. After all, in sparse data, a frequency of 3 or 4 might be sufficiently outstanding to be selected as a good candidate, and losing this information at the beginning of the undertaking might be severe.

Either way (via statistical significance indication or mere frequency), the result at the end of this stage is a structure containing the lexical (single word) candidates that have been found to be important in the context of the given input document. These lexical candidates will be used as input by the procedure extracing more complex units of text (chunks or n-grams), which will be described next.

4.2 Moving to Complex Units

Here, more complex units of words are composed and assembled in a list of *complex terms*. The previously yielded lexical candidates from *LexicalCandidateStore* are treated as cue tokens, in such a way that only complex units are considered which contain one or more *cue tokens*. Two different strategies for assembling complex terms have been implemented, making use of either *noun chunks* or *n-grams*. In case the linguistic annotation produced noun chunks in a previous step, the *noun chunk strategy* is employed as it is more accurate, otherwise the *n-gram strategy* is utilised. Both strategies are discussed subsequently.

4.2.1 Noun Chunk Strategy

Noun chunks containing one or more lexical candidates are considered for further processing, however they are transformed (or even rejected) according to a number of heuristics, a) to **avoid a larger degree of data sparseness** than necessary,
b) to **guard against possible errors made by preceding linguistic components** and
c) to **prune garbage or junk** that has been introduced as part of textual artifacts (mostly non alpha-numeric characters such as `[`, `"`, `/`, `(`, etc.).

The heuristics can be classified i) as *constraints and transformations on the token level*, controlling the assembly of the chunk, and ii) as *restrictions on the chunk level* to determine whether to cancel an assembled chunk or not.

Token level: As mentioned in 3.3.2, in most contexts, nouns are introduced by determiners (*the new shoes*) or pronouns (*her new shoes*), which, when counting observations over noun chunks, leads to an increase in observable types, at the expense of frequency counts. The result is sparse data which is problematic to interpret. [5] Therefore, the following constraints are imposed on tokens when assembling a chunk, and non-conforming tokens are not being used as building blocks. **Chunk-initial tokens must not be**

- **a numeral or number**
- **a determiner or pronoun**
- **a stopword**
- **a non-word** to filter out punctuation symbols.

Furthermore, a number of restrictions have been introduced, constraining the tokens in chunk-final position, mostly correcting a slight overgeneration of the noun chunker. Again, tokens not adhering to the following rules are not included into the chunk under construction: **Chunk-final tokens must not be**

- **an adverb**
- **a stopword**
- **a determiner or pronoun**
- **a non-word** to filter out punctuation symbols.

Elsewhere in Automatic Term Extraction, harder morphosyntactic constraints on the formation of extraction candidates (such as fixed sequences of part-of-speech patterns) have been imposed, most notably by [29, 28], where only noun-final chunks are considered, and [76], who only extract preposition-noun-verb combinations (PNV). This greatly decreases the amount of chunk candidates, a side-effect that is particularly harmful when dealing with small amounts of data. Additionally, a negative side effect from a multilingual point of view is introduced, as languages vary in their syntactic composition:

English		French		German
military	*base*	*base*	*militaire*	*Militärbasis*
ADJ	NOUN	NOUN	ADJ	NOUN

[5] It could be argued that the frequency should be counted over the head-nouns only, however unfortunately head-noun information was not available at the time of implementation and would have required additional effort.

Settling on dedicated extraction patterns would mean enumerating acceptable compositions for each supported language, which tends to become unmanageable and inflexible with an increasing number of languages. Hence, to allow for flexible cross-language use, morphosyntactic restrictions are kept at a minimal, universal level, allowing complex terms composed of any combination of content words (noun, adjective, verb, adverb), transferring the responsibility of promoting and demoting candidates to the confidence scoring functions described later in this chapter. Also performed on token level is a token-repair-procedure, with the aim of gluing together words that have been separated by hyphenation.

Chunk level: All of the constraints imposed on chunk level are for the purpose of pruning / preventing an assembled chunk to be carried on, largely addressing imperfect input such as junk characters or errors made by preceding components. A chunk is not further considered if it does not satisfy all of the restrictions introduced here.

They include the requirement to be **composed of a minimal number of characters** (2, by default) for each word contained in a chunk, to bypass orphaned letters or characters. Furthermore, only **an extended set of alpha-numeric characters is accepted**, consisting of letters, digits, whitespace and dash – also specified by the following Perl regular expression `/\w\s\d-/` – to properly handle string artifacts such as `(`, `/`, `]`, and so on. Finally, the scenario might arise where a preceding linguistic component made a mistake, having missed a sentence boundary which led to the erroneous assignment part-of-speech tags, resulting in unreasonably lengthy noun chunks. This error is predominantly encountered when processing web pages from the internet, where for instance the HTML is being stripped from a menu structure, resulting in a number of consecutive nouns which are processed as one sentence. The outcome is often a candidate similar to *Search term Explore the BBC BBC News Updated every minute.* To guard against such error types, noun chunks are **restricted in the amount of tokens they can carry** [6], to a maximum amount of 5 tokens.

As the various lookups between different levels of annotations in the GATE-internal document data structure resulted in an unacceptable speed penalty, look-up has been

[6] From a linguistic perspective this might cause raising eyebrows as even noun chunks can go on indefinitely: *Peter's expensive new big ... red shiny car*, cf. pumping lemma, Hopcroft & Ullman, chapter 3 [36], however it has been found a pragmatic solution here.

re-implemented using a number of hash tables. The outcome was a significant performance gain so that a look-up could be achieved which was up to 50 times faster than the naïve GATE-lookup.

4.2.2 Lacking Linguistic Resources: Fallback to N-Grams

In case no linguistic resource for *noun chunking* is available, an *n-gram strategy* is automatically selected for constructing units of more complex shape. Here, n-grams of tokens/lemmas containing one or more *cue tokens* are generated, varying in size from n=2 up to n=4. Essentially, the constructed n-grams must satisfy the same constraints introduced for the *chunk strategy*, and the same type of transformations are carried out in order to keep the candidate set clean.

Either way (via *chunk strategy* or *n-gram strategy*), the result at the end of this stage is a list containing instances of the complex term candidates that have been extracted as described above.

4.2.3 Partitioning the Candidate Space by String Similarity

The challenge now is to divide these instances into *clusters*, such that each cluster holds instances that are more similar to each other than to any of the members of any other cluster. The similarity criterion that has been chosen and is used for comparison considers the *lemmas* of the *complex terms*, hence the similarity sought is *string-based*, which makes *string-similarity metrics* the appropriate tools to be applied here. Popular string-similarity metrics include the *Levenshtein Distance* [7], the *L1 Distance* [8] and *Dice's Coefficient* [9].

For similarity clustering, the *Monge-Elkan Distance*, modified for recursive matching [53] was chosen, as – in a survey and evaluation paper by Cohen et al [14] – it was found to be among the highest scoring metrics for the type of task at hand.

Monge & Elkan define their algorithm as field matching problem, where a string str_A to be compared with a second string str_B comprises so-called fields, which again may contain subfields, determined by designators such as comma or whitespace. A

[7] also Edit Distance, as it computes the number of edit-operations that are needed to transform a source string str_A to a target string str_B

[8] also Manhattan (or Block) Distance, as it can be described using a grid in 2-dimensional space, thereby resembling the street layout of Manhattan

[9] here, similarity between two terms is defined as the number of strings common to both divided by the sum of strings in both terms

scoring function between two strings takes into account the amount of subfields in str_A that are maximally similar to subfields in str_B, recursively applying a matching function, thereby considering the semantic similarity of corresponding subfields, such that

$$sim(str_A, str_B) = \frac{1}{|str_A|} \sum_{i=1}^{|str_A|} \max_{j=1}^{|str_B|} sim(str_{Ai}, str_{Bj})$$

The achievement here is a proximity degree between str_A and str_B within the range of 0 and 1, reflecting the amount of overlapping subfields and atomic strings, resulting in higher confidence even for string pairs where Levenshtein assigns a low score. Naturally, the outcome of the Levenshtein Distance calculation is an integer, but – for ease of comparison – here it is scaled to a decimal number between 0 and 1 by

$$1 - \frac{levenshtein(str_A, str_B)}{max(length(str_A), length(str_B))}$$

as shown in the table below:

String A	String B	Monge-Elkan	Levenshtein
arithmetical computation	complex arithmetical computations	1.000	0.727
intertial frame	local intertial frames	1.000	0.682
pure gravitational field	homogeneous gravitational field	0.800	0.677

Here, a recursive version of the Monge-Elkan algorithm is used, and complex terms which exceed an empirically determined similarity threshold of 0.8 are grouped into the same cluster, using the SecondString package/library [10] developed by Cohen and colleagues from Carnegie Mellon University.

It could be argued that suffix tries are the better choice for this clustering procedure, as they show more potential in grouping phrases by their head-noun, which could lead to a low-hanging crop in terms of yielding a hierarchical structure. However, using suffix tries with this motivation fails for languages which do not necessarily exhibit a noun-final-position in noun chunks, such as French. As constructing hierarchical structures was not the concern of this undertaking in the first place, the course of action pursued here was mainly driven by pragmatic decisions, and using string metrics seemed a reasonable choice.

The comparison procedure over the list of complex terms L_{ct} has been implemented to run in $\theta(n) = n \cdot log(n)$ time, where $n = |L_{ct}|$, avoiding quadratic runtime.

[10] `http://secondstring.sourceforge.net/` – retrieved 2008-07-29

4.2.4 Folding a Cluster and Selection of its Representative

Now that the chunk candidates have been partitioned into clusters (or groups), it is important to stress that they are **similar, but not equal**. A cluster may consist of such different complex terms as ⟨*Central America, northern Central America, Central America, Central America, lower Central America, Central America*⟩ or ⟨*gravity field free, gravitational field, gravitational field, homogeneous gravitational field, pure gravitational field, gravitational field, gravitational field, gravitational field*⟩ where items may occur more than once, reflecting their appearance in the document.

So now the question arises how to represent this diversity in a meaningful way. It seems reasonable to select one element from the cluster that *best represents the other elements*, a so-called *representative*. As the concern here is key**phrase** extraction, priority should be given to terms consisting of more than one word – otherwise, the standard corpus linguistic methods which are working well to identify relevant single words – as elaborated on in section 2.3 could simply be applied. To achieve this, a *similarity maximisation* is calculated, where a cluster is represented as the set C of *complex terms* ct: $C = ct_0, \cdots, ct_{n-1}$. To find the complex term ct_{max} that resembles the remaining elements most closely, a pairwise string metric comparison [11] over the elements of the cluster is performed, competitively maximising similarity:

$$\forall ct \in C : msim_i = \sum_{j=0}^{n=|C|} sim(ct_i, ct_j)$$

However, this similarity maximisation aggressively selects short (one word) terms, as they are more likely to be similar to the mutual cluster elements. Here, it is important to repeat the invariant that the elements of a cluster already are similar. To promote a term ct_i made up of more than one word, a simple boosting of $msim_i$ by the number of words $|ct_i|$ is a first approximation, however this leads to an aggressive selection of the longest terms in a cluster, even when their similarity compared to their mutual elements is not so high, and some sort of control is needed: During the calculation of the intra-cluster similarities for ct_i, an observation o'_i is made, recording how often the outcome of the Monge-Elkan metric resulted above an empirically determined threshold of 0.8, reflecting very high correspondence. In combination with word count $|ct_i|$, it is now possible to control the similarity maximisation algorithm, and at the same time promoting multi-word phrases without over-estimating the importance of very long

[11] again, the Monge-Elkan distance is used, as described in the previous paragraph

phrases. The function

$$boost_i = log(|ct_i| \cdot o'_i)$$

was found to be suitable for boosting, and thus,

$$\forall ct \in C : msim_i = boost_i \cdot \sum_{j=0}^{n=|C|} sim(ct_i, ct_j)$$

and following, the complex term maximising the result is chosen as *representative* of the cluster:

$$ct_{rep} = max(msim_0, \ldots, msim_{n-1})$$

Central America : ⟨*Central America, northern Central America, Central America, Central America, lower Central America, Central America*⟩,
or

gravitational field : ⟨*gravity field free, gravitational field, gravitational field, homogeneous gravitational field, pure gravitational field, gravitational field, gravitational field, gravitational field*⟩

The *representative* yielded by this procedure is aggregated into the set of main *keyphrase candidates*. For these, a confidence score is being calculated, considering a number of features such as *significance* and *scope*, which is described next.

4.2.5 Regarding Positioning

Scope is one of the more important attributes of a term in relation to the document it appears in. Frequently appearing phrases might occur in a particular area of a document (e.g., in a book chapter), having *local scope* only, whereas phrases observed with a lower frequency might occur more evenly distributed over the whole document space, claiming *global scope* over the document. Also, phrases occurring from the very beginning of the document have a larger influence than phrases that are only observed in some section at the end.

Scope is implemented here as a function of *observations over document segments* (also called *partitions*), and the outcome is an assignment of either *global*, *local* or *none*, where phrases with assigned scope *none* are considered rare events, typically scattered over the whole document space such that there is no observation of relevance accumulated in one partition.

The following formula reflects the outlined considerations regarding scope over a document. It is computed for each representative r (as determined before) by firstly dividing the document into an ordered set P of n equally sized partitions, where n has been chosen as $n = 10$ for the implementation.

$$P = \{p_0, p_1, \ldots, p_{n-1}\}$$

Now,

1. For each partition $p_i \in P$

   (a) count the number of observations $obs(p_i)$ of cluster elements belonging to representative r contained in p_i

   (b) define a binary mapping from $obs \rightarrow obs'$ recording whether elements were observed at all in p_i:

$$obs'(p_i) = \begin{cases} 1 & \text{iff } obs(p_i) > 0 \\ 0 & \text{otherwise} \end{cases}$$

2. compute $g'(P)$, resulting in the binary value of 1 iff an observation was made ($obs'(p_i) = 1$) in half or more of the partitions, and at the same time, an observation was made more than once in the partitions representing the first half of the document ($|_{i=0}^{\lfloor \frac{n}{2} \rfloor} p_i$):

$$g'(P) = \begin{cases} 1 & \text{iff} \quad \sum_{i=0}^{n} obs'(p_i) \geq \lfloor \frac{i}{2} \rfloor \\ & \qquad \wedge \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} obs'(p_i) > 1 \\ 0 & \text{otherwise} \end{cases}$$

3. compute $l'(P)$, resulting in the binary value of 1 iff an observation was made in two consecutive partitions ($obs'(p) = 1$ for one pair $\langle p_i, p_{i+1} \rangle$), or at least one of the partitions has an observation of greater than 1 ($obs(p_i) > 1$):

$$l'(P) = \begin{cases} 1 & \text{iff} \quad \exists p_i \quad obs(p_i) > 1 \\ & \qquad \vee \quad (\exists p_i \quad [obs'(p_i) = 1 \wedge obs'(p_{i+1}) = 1]) \\ 0 & \text{otherwise} \end{cases}$$

4. use the previously computed values g' and l' for scope assignment:

$$scope(g', l') = \begin{cases} global & \text{if } g' = 1 \\ local & \text{if } g' = 0 \wedge l' = 1 \\ none & \text{otherwise} \end{cases}$$

Figure 4.3 illustrates the scope assignment process for a number of examples, e.g., "inertial frame" occurs often and fairly distributed over the whole document, thereby claiming global scope, whereas "gravitational field" satisfies the first option of the local scope requirement, and "identical clocks" satisfies the second option.

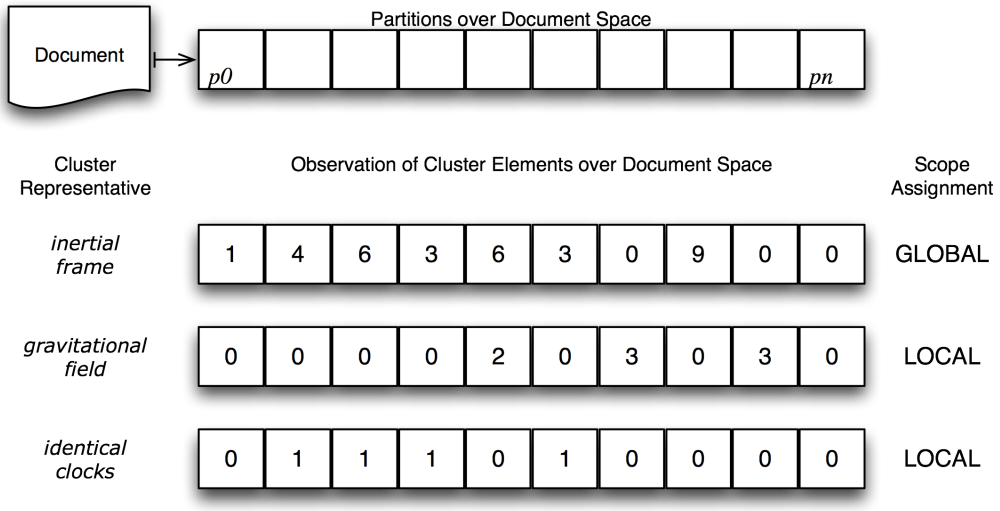

Figure 4.3: Scope Assignment Illustrated

The position analysis for each of the previously determined *representatives* addresses the scope factor, such that the findings can influence the *confidence scoring functions* outlined in the next section.

4.3 Ranking Keyphrase Candidates

Now that a number of factors influencing the degree of descriptiveness have been isolated and can be programmatically determined, it is time to combine them into a meaningful, singular value which can be used for an *overall ranking* of the selected *representatives*.

The here engineered scoring function takes into account three different aspects, each contributing with a dedicated, carefully selected weight such that the overall confidence

score remains within the boundary of zero and one.

The aspects focused on here are

1. *significance of cue tokens* in the representing candidate: CtS
2. *scope*, as determined by the distribution of the candidate cluster over the document: SoC
3. *number of words* contained in the candidate / representative: NoW

Their combination results in a confidence score for a representative / keyphrase candidate

$$conf = CtS + SoC + NoW$$

Subsequently, each of the contributing aspects are motivated, defined and briefly discussed:

4.3.1 Cue-token Significance in Representing Candidate (CtS)

Here, the *significance* (or strength) of the *cue token(s)* contained in the representative is considered, as it was assigned at the end of the lexical analysis. This could either be a χ^2 score, or the mere frequency in case no statistical computation was made for the reasons mentioned in section 4.1. As frequency of the cue token influences the χ^2 score to some extent, it is possible to say that here, it is indirectly incorporated. The outcome of this particular contribution should achieve the following:

- **penalise** candidates containing a weak cue token significance
- **promote** candidates containing highly significant cue tokens
- **expected result**: a numerical contribution reflecting the importance of the contained cue token in a candidate

With these requirements specified, let $weight_{CtS} = 0.3$ be the scaling constant,

$$\text{and } sig'(ct) = \begin{cases} sig(ct) & \text{if a significance value was assigned for } ct \\ scale \cdot freq(ct) & \text{otherwise, where } scale = 10 \end{cases}$$

Note that the frequency value is scaled by 10 in case no significance value was assigned. The most important case of an occasion is the following: The cue token was not contained in the lemma list of the reference corpus, as a result χ^2 was not computed and pure frequency is carried on. The implication here is that if the cue

token is not represented in the reference list (which ranks only the 5,000 most frequent noun lemmas), it can be assumed that it will be found at a lower rank, resulting in a relatively high χ^2 score [12].

The significance contribution function CtS is defined as

$$CtS = weight_{CtS} - \frac{1}{1 + ln(1 + sig'(ct))}$$

Then, theoretically, CtS must fall within range [-0.7;0.3]. However, for negative values to be computed here, it would imply a confidence score well below significance threshold as described in 4.1. Typically, CtS will be observed within the range [0;0.18], as the selected logarithm-function (ln) wears off asymptotically, restricting the formula to assume larger values.

4.3.2 Scope Contribution of Candidate Cluster (SoC)

To properly control the influence of the *scope* of a cue token *ct* as discussed before, the following desiderata for this part of the formula have been derived: *global scope* shall have priority over *local scope*, and *local scope* shall have priority over *no scope*, such that a partial order wrt **scope priority** $sp(scope)$ can be given as

$$sp(global) \succ sp(local) \succ sp(none)$$

Thus, $sp(scope)$ is a simple mapping from *scope* to a decimal number within the range [0;1].

In case of small input documents, it is most likely to observe a larger proportion of phrases that have no scope. In anticipation of this scenario, the following policy is proposed:

- **neutral stance** towards phrases of scope *none* [13]
- **mild promotion** of phrases with assigned local scope
- **considerable promotion** of phrases showing global scope

Having discussed these issues, a formula was chosen, with the outcome of scaling a scope assignment to a value within the boundaries [0;0.5].

[12] of course, a typo could be responsible for the cue token not being represented by the reference list, however the typo would have to be systematic as a significant amount of observations have already been made, which is ensured by the selection of the appropriate lexical strategy

[13] i.e. which are scarcely distributed over document space

Let $weight_{SoC} = 0.5$ be the constant controlling the scale of the overall assignment, and $sp(scope)$ the mapping from scope to a decimal number

$$sp(scope) = \begin{cases} 1 & ct \text{ has global scope} \\ \frac{1}{2} & ct \text{ has local scope} \\ 0 & \text{otherwise} \end{cases}$$

as discussed above, then the final *scope contribution* is defined as

$$SoC = weight_{SoC} \cdot scope(ct)$$

4.3.3 Number of Words (NoW)

So far, statistical significance has only been determined for single words, leading to a list of interesting *lexical candidates*. To distinguish the approach and tailor it towards *prediction of multiword candidates*, a boosting is performed for those candidates consisting of more than one word. As commonly a single term is expected as prediction, those cases should not be penalised explicitly, but rather, on a number of occasions when a prediction happens to consist of multiple words, these ones should be promoted.

In this fashion, let $weight_{NoW} = 0.2$ be the constant in charge of the scale of the overall assignment, and $|r|$ the number of words contained in the representative r, then

$$NoW = weight_{NoW} - \frac{weight_{NoW}}{|r|}$$

Again, the choice of the constant $weight_{NoW}$ ensures the scaling to an upper limit, between [0;0.2].

As $|r| > 0$ always holds, the function maintains neutrality towards candidates consiting of one word only, and in other cases, mildly boosts candidates comprising of 2 words or more.

It shall be noted that neither of the three contribution subformulæ are orthogonal to each other, however they have been engineered in such a way that for each subformula, the individual aspect is commanding the outcome of the result, which allows for control on an abstract, yet fine-grained level, leading to a confidence score comparable to other candidates extracted from the same document.

4.4 Summary

The last two chapters covered the main contribution of this thesis. In chapter 3, the emphasis was on discussing a number of *use-cases*, and based on them, deriving *requirements* that influence the main *design decisions*. The chosen *framework*, GATE, was briefly introduced, and the implementation and orchestration of necessary *linguistic preprocessing steps* was outlined.

This chapter elaborated in detail on the various aspects and stages central to the actual *keyphrase extraction algorithm*. To summarise, the keyphrase candidate selection procedure is as follows: A statistical χ^2 measure is computed over *lemmas of lexical items* found in the document in relation to a *large, general reference corpus* to assess their significance in the context of the given document. The top 25% (or top 25, whichever comes first) significant lemmas of the document are considered *cue tokens*, and noun chunks (or n-grams, in absence of a noun chunker) containing them are extracted and used to construct *complex terms*. Subsequently, the list of *complex terms* is partitioned into clusters by the Monge-Elkan string-similarity metric, in order to form *clusters* containing *similar complex terms*. For each cluster, one *representative* is determined, exhibiting the maximal boosted *intra-cluster similarity*. This representative corresponds to a *keyphrase candidate*, and its scope over the document is assessed. A *confidence-scoring* function is computed for each candidate, relying on *significance of cue token* contained, *scope contribution* of representing cluster, and *number of words* of the candidate. The result is a *ranked list of keyphrase candidates*, each one with an assigned *confidence score* between zero and one.

In the following chapter, the experiments that have been conducted to evaluate the algorithm will be described.

Chapter 5

Evaluation

> *If your experiment needs statistics, you ought to have done a better experiment.*
>
> Ernest Rutherford, British Chemist, Nobel Prize in 1908, 1871–1937

The literature on the evaluation of keyword extraction systems for single documents has been scarce, and little has been reported on quantitative assessment. This is reflected for instance in [3], [67] and [12], where only qualitative evaluation by conducting user studies is reported. The problem is partly related to the difficulty of applying proper evaluation metrics, as it is debatable whether the classic metrics commonly used in Information Retrieval such as *recall* and *precision* apply. Researchers who tried to provide a quantitative evaluation are struck with low precision as a result of a complete gold standard, as for instance pointed out in [75] and [38]. Those metrics would imply a dataset to be used as *gold standard* (or *ground truth*) which clearly provides a number of items for each document that are marked as *true positives* and *true negatives*.

Acknowledging the importance to assess the performance in a **quantitative** and **qualitative** way, this chapter reports on the attempt to perform both types of evaluation, and the results obtained. As Bourigault points out, an automatic extraction system cannot be expected to provide perfect precision (it will necessarily overgenerate), and although his study was carried out in Automatic Term Extraction (ATE), his observation applies equally here:

> *It is not possible to expect [a] program to extract terminological units* and nothing else*, given the basically referential semantic function of occurrences of terminological units: this means that the results obtained can only be considered, a priori, as* likely *terminological units.* [5]

Besides assessing the performance of the system, a very strong emphasis of the evaluation is to identify the *major drawbacks* of the approach in order to be able to address them at a later development cycle.

5.1 Strategy

The evaluation of the keyphrase extraction algorithm is divided into a quantitative assessment performed on a dataset obtained from PubMed Central, and a qualitative assessment in form of a user study. In the following, the rationale behind those choices will be motivated.

5.1.1 A Highly Subjective Task

The extraction of descriptive terms from a document is a highly subjective task, as Maynard comments [50]. Different persons have diverse mental representations of a given document, depending on their domain, the depth of knowledge of the document in question, and their attitude/stance towards its content. In this respect, the extraction of keyphrase candidates very much resembles the characteristics of a learning problem, as the distinction of "right" and "wrong" can barely be described in absolute terms, and the notion of correctness is rather defined on a gradual scale. This problematic attribute of evaluating learning problems has been noted before, and Brewster et al point out:

> *Precision and recall depend on a clear set of items concerned, for example Parts of Speech. There is no clear set of 'knowledge to be acquired' because the same set of facts can give rise to very different interpretations and therefore different kinds of 'knowledge'.* [7]

An evaluation of the keyphrase extraction algorithm based solely on objective measures is meaningful only to some extent, partly because the question arises what actually constitutes a gold standard, and partly because it is debatable whether such an experiment can be set up as a selection experiment for human judges. The task may be too subjective for subjects to end up with a sufficient amount of inter-annotator agreement over selected keywords/keyphrases (Barker & Cornacchia report on spectacularly low kappa values [3]), an issue also encountered in gold standard annotations for the GENETAG dataset [73].

In order to account for this, the evaluation procedure has been carefully set up into a *quantitative assessment* and a *user study*, as reported next.

5.2 Quantitative Evaluation

In order to be able to conduct a meaningful quantitative evaluation of acceptable scale on document level, it was necessary to obtain a gold standard/ground truth that has keywords attached to its documents. The open digital archive PubMed Central [1] currently hosts literature from 644 different journals of the biomedical domain and the life-sciences. A subset of its literature is provided as an XML dataset [2] for NLP and data mining purposes, which has been utilised for construction of the *ground truth*. The following will describe the procedure of creating a *gold standard* against which the qualitative evaluation can be performed.

5.2.1 Experimental Setup & Dataset Preparation

The dataset obtained from PubMed Central [3] comprises 77,496 peer-reviewed articles. Its XML schema offers a fine grained distinction of various aspects of publication-related metadata (journal, authors, affiliation, index-terms/keywords), full text and references. From these articles, only those containing assigned keywords were considered, reducing the number of articles used as evaluation dataset to 1,323, consisting of 4,921,583 words in total. The 1,323 articles were distributed across 254 different journals published by PubMed Central, ranging from *Abdominal Imaging* to *World Journal of Urology*. The text entered into the keyphrase extraction algorithm did not contain the keyphrases assigned to the articles. The workflow of the dataset construction and quantitative evaluation is given in figure 5.1.

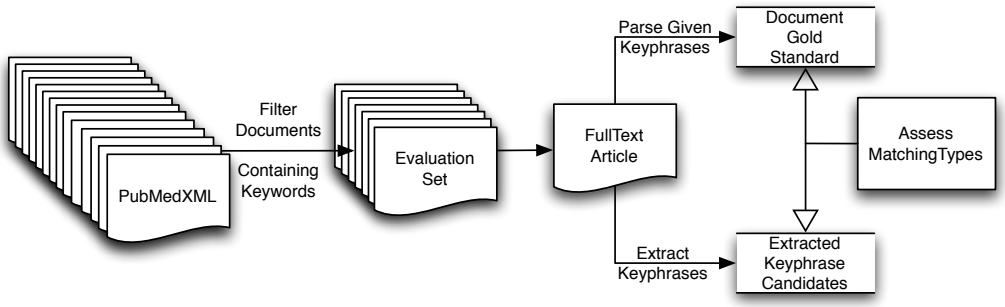

Figure 5.1: Quantitative Evaluation Workflow

[1] `http://www.pubmedcentral.nih.gov` – retrieved 2008-06-30

[2] `http://www.pubmedcentral.nih.gov/about/ftp.html#XML_for_Data_Mining` – retrieved 2008-06-30

[3] `ftp://ftp.ncbi.nlm.nih.gov/pub/pmc/articles.tar.gz` – retrieved 2008-06-30

Average document length was at 3,720 words (median: 3,473 words), with the longest document at 15,782 words, and the shortest document at 6(!) words. The total number of assigned keywords in this dataset was 6,931, such that the average mean of assigned keywords/keyphrases per document was just above 5, with the median at 5 as well. The maximal number of keyphrases assigned *a priori* to a document was 29, the minimum was 1. Document size and amount of assigned keyphrases were not found to be corresponding to each other.

5.2.2 Considerations & Metrics

Before settling upon evaluation metrics, it is important to consider the scenarios that may be encountered when assessing the predictions of the algorithm against the gold standard. A naïve approach would consider the gold standard as non-forgiving ground truth, that only permits predictions that fully match to be counted as positives, whereas predictions matching partly would be regarded as negatives (or errors). Other approaches to keyphrase extraction have relaxed the matching criterion in their evaluations by stemming **both** the gold standard terms **and** the predictions with the Lovins [44] stemmer before assessment [75, 27].

Often it is the case that partial matches are too close to the gold standard to be treated as error, as very commonly encountered in Biomedical NLP scenarios [25, 73]. The following introduces such types of partial matches by typical examples found in the dataset and discusses their proper treatment.

The Perfect Match, Partial Hits & Misses

Here, matching is done ignoring upper and lower case. It shall be noted that a match is only considered as such in case of word containment of some sort, namely when a gold standard term contains the predicted term or vice versa, ruling out false partial matches of the type *fascinating* – *fascism* [4]. Although this strategy permits direct antagonists such as *fascism* and *anti-fascism*, this is only valid as it is hard to argue against the fact that the first term has semantically nothing in common with the second term.

The following discussion of partial matches is also depicted in figure 5.2, and corresponds fairly to match criteria surveyed in [70].

Full Match: A full match is a *one-to-one mapping* between a predicted term and a term contained in the gold standard for a respective document.

[4] a more absurd example can be found in German: *Fasching* vs. *Faschismus* (*carnival* vs. *fascism*)

Source	Example
Gold Standard	Cerebral cortex
Prediction	cerebral cortex

Class1 SuffixMatch (Right-Boundary Match): Partial matches of this type share the common suffix, which in the vast majority of the cases is the head noun of the term. Thus, these matches are considered as very important. The mismatch here is the modifier which is lacking in the prediction, resulting in an inclusion of the predicted term in the gold standard term. Semantically, the gold standard term is more specific than the predicted term.

Source	Example 1	Example 2
Gold Standard	Traumatic brain injury	Head and neck cancer
Prediction	brain injury	neck cancer

Class2 SuffixMatch (Right-Boundary Match): These partial matches show the opposite inclusion characterstics compared to *Class1 SuffixMatches*, as here it is the gold standard term which is contained in the predicted term, but still sharing the suffix. Mostly, this implies the prediction being more specific than the gold standard term.

Source	Example 1	Example 2
Gold Standard	Epilepsy	Lymphoma
Prediction	temporal lobe epilepsy	Bcell lymphoma

Class1 PrefixMatch (Left-Boundary Match): In this case, often a singular form is predicted where a plural form constitutes the gold standard (example 1). However, another – slightly less frequently observed – partial match is the first part of a compound noun, lacking the head noun (example 2). While the first case nearly is a full match, the second case is more problematic, as the head noun (in most western languages) assigns the semantics to a noun phrase, resulting in different meanings of the terms in example 2 — the prediction is solely about the *cell cycle*, whereas the gold standard term is about *its control*.

Source	Example 1	Example 2
Gold Standard	Distal radial fractures	Cell cycle control
Prediction	distal radial fracture	cell cycle

Class2 PrefixMatch (Left-Boundary Match): Matches of this sort show a similar characteristic as the relation between Class1 and Class2 SuffixMatches: here, typically the ground truth term is singular whereas a plural term is predicted, as seen in example 1. Example 2 however shows a variation of the gold standard term is predicted, resulting in this type of partial match.

Source	Example 1	Example 2
Gold Standard	Biological agent	Haloarchaea
Prediction	biological agents	haloarchaeal genomes

The introduced differences in partial matching are graphically displayed in figure 5.2, and can be characterised in the contingency matrix of table 5.1. As discussed, Class1

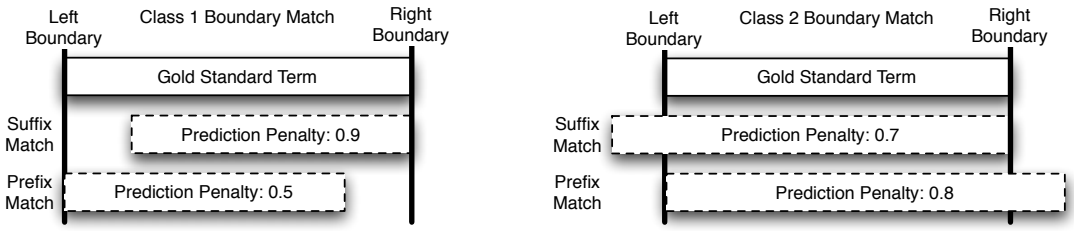

Figure 5.2: Boundary Matching Types

suffix matches typically widen the scope of the meaning conveyed by the target term, thereby deviating very little (and only in a more general way), which should be reflected in a very low penalty when deciding the grade of correctness, as in the context of the whole set of predictions, the more generic phrases are grounded again. Also, typical Class2 errors are still very close to the target and should only be slightly penalised, whereas Class1 prefix matches often imply a slight semantic transfer in case of a lacking head noun compared to the gold standard. While such a deviation is not desireable, the proximity of the predicted term to the gold standard term is still close, and thus it should be reflected in a medium penalty.

	Prefix (Left Boundary)	Suffix (Right Boundary)
Class1	Semantic Transfer *medium penalty*	Widening Scope *very low penalty*
Class2	Syntactic Variation *low penalty*	Narrowing Scope *low penalty*

Table 5.1: Contingency Matrix for Partial Matching Classes

This scoring philosophy is closely related to the one used in GENETAG evaluation efforts as reported in [70] and to the evaluation scheme for Named Entity Recognition used by the Message Understanding Conference (MUC-4) [5] in 1995.

Now, with these considerations in mind, it is possible to define a partial order over the previously introduced matching classes as depicted in figure 5.3, which also reflects the degree of correctness when evaluating for *recall*, where 1.0 means a full match, and 0.0 means no match at all:

FullMatch		C1Suffix		C2Prefix		C2Suffix		C1Prefix		NoMatch
1.0	$\succ$	0.9	$\succ$	0.8	$\succ$	0.7	$\succ$	0.5	$\succ$	0.0

Figure 5.3: Partial Order over Partial Matching Classes

If Recall Only Means so Much, What About Precision?

The previously defined order offers a reasonable instrument for applying the standard information retrieval metrics *recall* and *precision* in a **fair evaluation**, despite the problems pointed out at the beginning of this section.

Recall is given as the correlation of the amount of correctly predicted keyphrases and the amount of keyphrases specified by the gold standard:

$$recall = \frac{\#\text{ of correctly predicted keyphrases}}{\#\text{ of gold standard keyphrases}}$$

Precision is defined by the correlation between the amount of correctly predicted keyphrases and the number of total predictions:

$$precision = \frac{\#\text{ of correctly predicted keyphrases}}{\#\text{ of total predictions}}$$

To put the fair evaluation into a context, two additional assessments were considered, (i) a **strict evaluation**, treating all **partial matches as negatives** (or errors), and (ii) a **relaxed evaluation**, analysing all **partial matches as positives** .

It is important to note that the introduction of partial matching also opens the door for *ambiguity*: In a document, multiple candidates could be mapped to the same single target keyphrase with the same matching type. For instance candidate *ac* and candidate *bc* could both be assigned a Class1 SuffixMatch for gold standard phrase *cc*.

[5] `http://www.aclweb.org/anthology-new/M/M92/M92-1002.pdf` – retrieved 208-07-22 — a description of the MUC-4 evaluation scheme

In such a case, in order to maintain correct counts for positives and negatives, only the candidate with the highest confidence value is considered a true bearer of such a matching type, rendering the other equally matching candidates to NoMatch.

At the same time, a candidate *cc* could also be found for target *cc*. In such a case, preference is given to the match type with higher priority, which is derived from the defined partial order, such that FullMatch overrides Class1 SuffixMatch, and as a result, only the highest match type retains its status, whereas the disfavoured candidate will be assigned a NoMatch in turn. This ensures that **at maximum one true positive per gold standard item is being accounted for** when it comes to calculating precision and recall.

5.2.3 Results & Discussion

For the 1,323 documents, the total amount of generated keyphrase predictions was 52,825. The maximum number of keyphrase predictions for a document was 99, and the minimum number was 0 (on 13 occasions, for very short documents of average length 64 words / median 29 words). On average, just under 40 keyphrases were extracted per document, with median at 40.

	Distribution	Prediction Ratio	Target Ratio
FullMatch	2065	0.039	0.298
C1Suffix	959	0.018	0.138
C1Prefix	517	0.010	0.075
C2Prefix	322	0.006	0.046
C2Suffix	204	0.004	0.029
NoMatch	47243	n/a	0.413
Prediction Total	52825	(Target Total: 6931)	

Table 5.2: Distribution of Matching Types in Fair Evaluation

Table 5.2 displays the matching distribution for the **fair evaluation**, which are the main results obtained for this quantitative evaluation, and referred to unless otherwise stated. Prediction ratio gives a view on the matching distribution relative to the total number of predictions, whereas target ratio displays the matching distribution relative to the gold standard. The actual precision and recall values for strict, fair and relaxed assessment can be derived directly therefrom, and are displayed in table 5.3.

Given these values, the first thing to note is that precision is very low. The problem faced here is the large amount of keyphrase predictions constituting the NoMatch set. As mentioned above, on average 40 predictions are competing for only assigned 5 target

	Precision	Recall
Fair	0.077	0.518
Strict	0.039	0.298
Relaxed	0.089	0.587

Table 5.3: Fair, Strict & Relaxed Evaluation

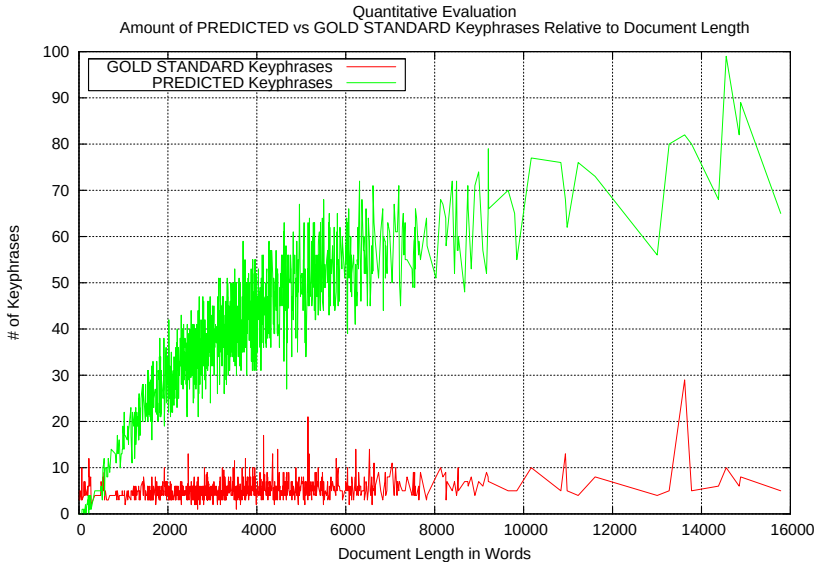

Figure 5.4: Assigned vs Predicted Keyphrases Relative to Document Length

keyphrases, and looking at figure 5.4 reveals that the amount of predictions increases dynamically with document size, whereas the number of *a priori* assigned keyphrases constituting the gold standard remains more or less stable around 5 regardless of document length.

It shall clearly be noted that instead of restricting the result set to a number of n-best items as commonly observed ([75], [12]), here **all identified keyphrase candidates are presented to the judges**, offering a larger degree of vulnerability for a decline in precision.

Moreover, Turney [75] and Jones et al [38] observe that not necessarily all *a priori* assigned keyphrases are actually contained in the document, as frequently recurring phrases in the text body may be rephrased for keyword assignment, empirically determining the average proportion of containment at roughly 75%. The implication of this observation is that a recall expectation of 100% is an unrealistic one. A good example from this evaluation is found in the article "Biomechanics of Traumatic Brain

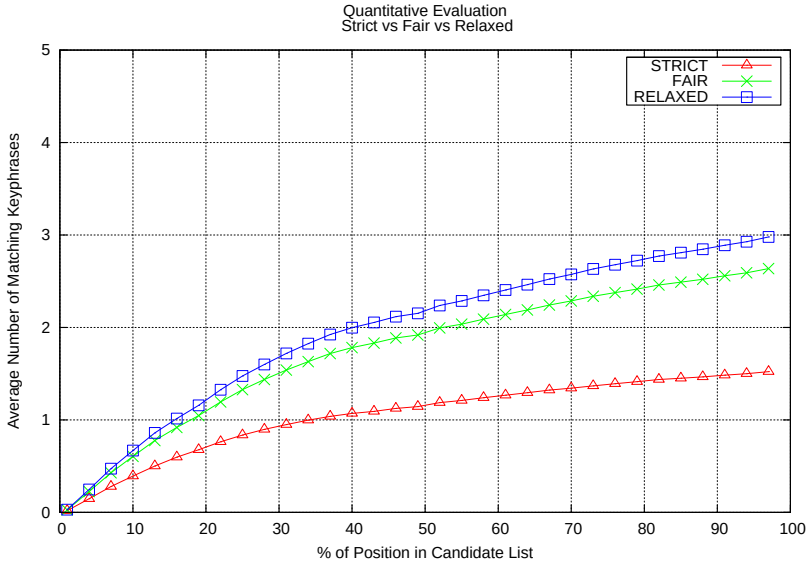

Figure 5.5: Keyphrase Matches in Candidate List

Injury: Influences of the Morphologic Heterogeneities of the Cerebral Cortex" (PubMedID 2413127) [6], where the index term "Inhomogeneities" is given, but never actually mentioned in the full text. Instead, "heterogeneities" is used quite often in the article, and extracted as a keyphrase candidate, unfortunately with a NoMatch result. [7]

Further focussing on the results obtained from the **fair** evaluation, figure 5.5 displays that on average, nearly 2 *a priori* keyphrases could be matched after the top-40% of the candidate list. Taking into consideration that the average number of assigned keyphrases is 5, and that possibly not all are containted in the document, the average recall with 51.8% does quite well when compared to the KEA evaluation, where recall on author assigned keyphrases is reported on average at 17% [38].

Eventually, an investigation into the distribution of matching types, and their contribution relative to the position in the candidate list (which is ordered by *confidence*) revealed that candidates of type FullMatch are commonly found at the top segments of the list, as figure 5.6 reveals. This is a positive sign, as it shows that the ranking function for score assignment – as elaborated on in section 4.3 – does its job reasonably well, accumulating the most contributing (and therefore, important) candidates at the beginning of the list. It is also notable that this finding is confirmed in the qualitative evaluation results discussed in the next section.

[6] `http://www.pubmedcentral.nih.gov/articlerender.fcgi?tool=pmcentrez&artid=2413127` – retrieved 2008-08-14

[7] It was quite easy to find this very example by manual inspection, which could simply be bad luck, however the suspicion remains that a non-neglectable proportion of assigned keyphrases are non-retrievable from the full text.

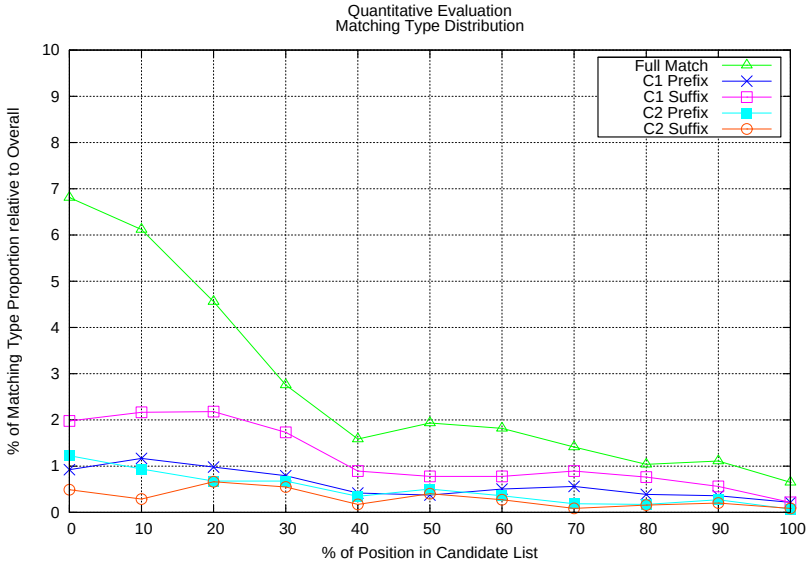

Figure 5.6: Matching Type Distribution

Figure 5.6 also exposes that the impact of the partial matching types is low throughout the prediction space, except for the C1 SuffixMatch, which also is a major contributor and stands out when considering partial matches only, a fact that will be interesting to look back to when examining the results of the user study.

5.3 Qualitative Evaluation

As reported in the last section, a quantitative evaluation – even if performed in a fair manner – only yields a lower estimate of the system performance as it is gold-standard centric. In this case however, the gold-standard can only be seen as a weak approximation of the truth. Most likely, more terms exist to describe a scientific article of several thousand words than the 5 keyphrases assigned to it on average. While partial matches can – to some extent – be handled quantitatively in a meaningful manner, it is incredibly difficult to programmatically gauge the goodness of predictions that are not even partly contained in the gold standard. The ground truth only includes positives, whereas negatives are not given, and the assumption that every non-positive automatically should be treated as negative is unrealistic in the context of keyphrase extraction, where the *quality of keyness* for a term is more of *gradual* matter than a strict binary classification. At the same time, the grade of keyness assigned to a given candidate may vary wildly when considering different judges. This again, leaves the evaluation in a debatable state, which is precisely what any form of assessment aims

to avoid in the first place.

Therefore, a user study was conducted, where the self-selected judges are preferably authors of the documents used as evaluation dataset. As the judges were free to choose documents of their liking [8], inter-annotator agreement such as Kappa [10] – which seemed to be problematic for this kind of task in previous experiments [3] – became irrelevant for this qualitative evaluation. Furthermore, as human interaction takes place in most use-cases where the tool has been deployed, this form of user study resembles a real-world scenario more closely. Next, the considerations underlying the experiment are outlined, and the path taken to conduct the user study are described.

5.3.1 Setup & Experiment

Each judge taking part in the experiment was asked to provide the system with a freely chosen number of documents he knows very well, preferably documents that have been (co-)authored by him. It is the judge's task to accept or reject any predicted keyphrase for the documents presented to the system, where in case of a rejection, the details for the reason of rejection were sought, from a set of 3 options:

too general : predictions that are too general to have distinctive character for the document

too specific : predictions that are too specific to make a statement about the document as a whole

nonsense : predictions that are not related to the content of the document, because they are not relevant, or because they constitute mistakes accumulated by the linguistic preprocessors during candidate formation

Besides giving a more realistic view on precision and user-acceptance, the experiment is also regarded as insightful for possible adjustments of the system in a future development cycle, particularly when considering the distribution of reasons for rejection.

The experiment was conducted via a web-application specifically developed for this undertaking, presenting the self-selected judges with an interface to the keyphrase extractor and an upload mechanism for the documents. The predictions were also presented via a web-interface in such a way that judges were able to conveniently fill out the generated forms and submit their assessment, as depicted in figure 5.7. The full text of the instructions for the experiment is given in appendix C.

[8] modulo some restrictions for language, file format and some suggestions regarding size

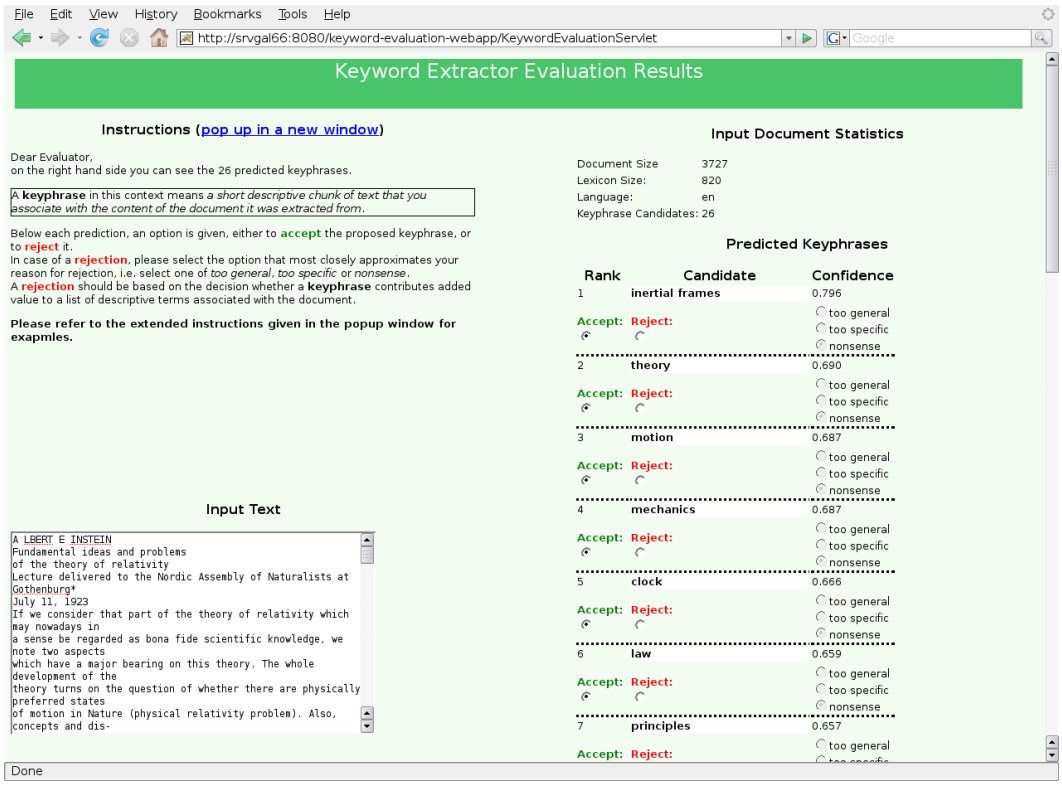

Figure 5.7: Evaluation Form for Qualitative Assessment of Keyword Extraction

5.3.2 Results & Discussion

Overall, 47 users signed up for the experiment, which was running for 10 days. In total 94 documents were used as input, with the largest document at 81,668 words, whereas the smallest document consisted of only 4 words [9]. The average document length was 7,671 words, the median was determined at 5,128 words per document, and it took an average of just over 3 minutes to determine the "good" and "bad" candidates per user, per document.

No restrictions were imposed on the document content, and judges were encouraged to use documents from all sorts of domains, ranging from scientific articles, technical records such as RFCs, contemporary writing and news, to personal communication. Documents however were required to be written in English, and of type PDF, Microsoft Word, plain text or HTML. The judges came from a multitude of backgrounds, ranging from PhD students and researchers (mostly in computer science) to IT professionals, engineers, as well as persons employed in the financial sector.

An overview of the evaluation runs per judge is given in table 5.4(i), and table 5.4(ii)

[9] the instructions suggested to use reasonably sized documents, if possible consisting of at least 500 words

shows a breakdown of the average acceptance ratio per document length, which has been partitioned into segments of individual size. Table 5.5(i) gives the overall scores

Judges	Documents
1	10
1	9
1	8
2	4
3	3
13	2
27	1
Total: 47	94

Size in Words	Documents	Accept %
0 – 500	8	41.8
501 – 1,000	7	49.3
1,001 – 2,000	10	42.6
2,001 – 10,000	55	49.4
10,001 – 25,000	8	46.8
25,001 – 50,000	3	32.4
50,001 – 100,000	3	70.6
Total: 721,157	94	

Table 5.4: Breakdown of (i) Evaluation Runs per Judge, (ii) Document Length

for accept and reject assessment, stating that acceptance with 49% almost matches the amount of rejection. This finding is about 13% below from what has reported by Turney [75], however in his experiment only top-7 keyphrases are presented to the judges, whereas in the experiment described here **all identified keyphrase candidates are considered**, without postprocessing them and excluding phrases below a certain threshold or rank, which is also partly responsible for the increase of the candidate set at increased document length.

Most evaluation runs were performed on documents between 2,001 and 10,000 words, as table 5.4(ii) indicates, which corresponds to the size of average scientific conference articles. In this segment, *accept ratio* was with 49.4% almost even up with reject ratio. Instructions suggested not to use documents with less than 500 words, nevertheless this was the case on 8 occasions. Figure 5.8 presents a view on evaluation

	%	Absolute
Accept	49.0	1,273
Reject	51.0	1,327
Total	100.0	2600

	Reject %	Absolute
too general	52.2	693
too specific	13.6	181
nonsense	34.1	453

Table 5.5: (i) Qualitative Evaluation Result, (ii) Breakdown of Reasons for Rejection

runs, documenting the amount of overall acceptance per run, over segments of 10%. As can be seen, most runs were completed with an acceptance ratio between 20% and 60%, altogether accumulating the major distribution of runs with 69%.

Now, focusing on the document length, and segments between 501 and 50,000 words, which have accumulated most evaluation runs, a slight trend is observable. Accept and reject votes are in close proximity near 50%, with rejection slightly superior, as also

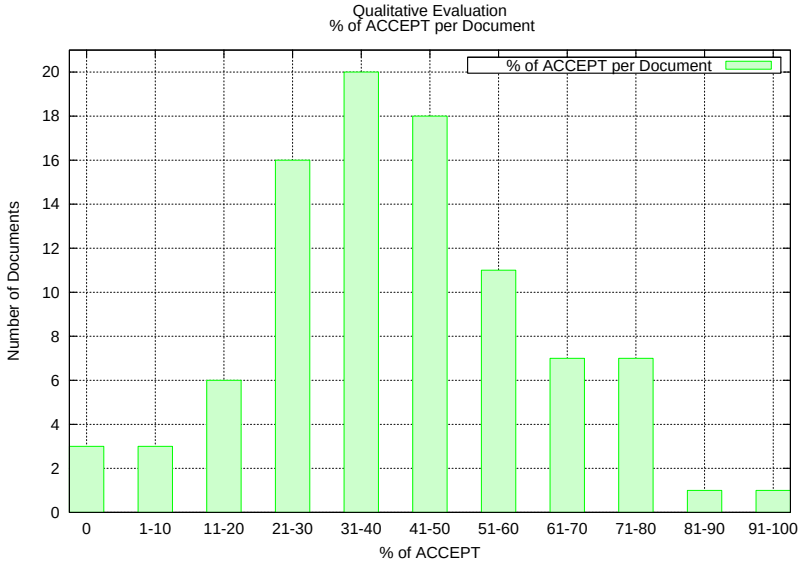

Figure 5.8: Document/Evaluation Run Distribution over Acceptance

depicted in figure 5.9, except for the segment denoting documents of 1,001–2,000 words, where acceptance declines more, to 42.6%. An inspection of the respective segment gave no concluding answers, and a comparison with the next smaller segment (501–1,000 words) – as documents in both segments are likely to have similar character – only revealed that the amount of rejections classified as *nonsense* by the judges was twice as high in the 1,001–2,000 section.

A more detailed analysis revealed that out of 39 rejections of class *nonsense*, 18 were due to text conversion problems or mistakes in the linguistic preprocessing stage:
(a) On 3 occasions, hyphenation from PDF files could not be undone, and ligatures were not correctly converted to letters (i.e. fi ↦ `fi`, ff ↦ `ff` did not succeed, resulting in unknown or garbled words such as `le-based o ine editing` from originally "file-based offline editing").
(b) Sentence boundary mismatches were responsible for mistakes on 5 occasions, as in many scientific articles, the ending of a section heading is not marked by a sentence delimiter, and as such not recognised due to formatting loss. The following sentence, which begins directly after the heading, is treated as the continuation of the sentence that was started by the heading itself, affecting the noun chunker as a result, leading to examples such as "Database Representation The database representation".
(c) Textual artifacts from tables, figure captions or formulae in tables/captions contributed 5 times as did artifacts from non-coherent text (such as text for menu structure in web pages).

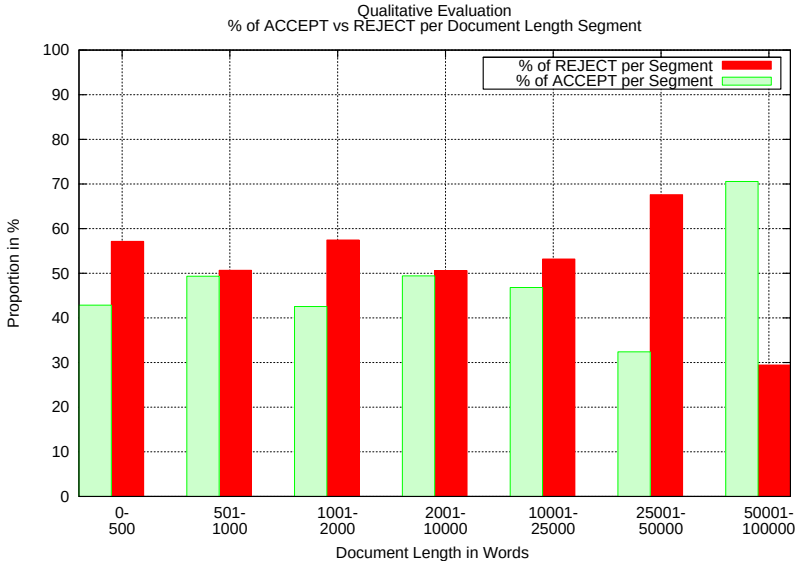

Figure 5.9: Breakdown of Accept Distribution per Document Length

Other segments were not affected that much by such problems, however, inspecting the overall *nonsense-based rejections* uncovered that the proportion of the above mentioned errors was about 28%, from a total of 453, a discovery that will be revisited in chapter 5.4.

Looking more closely at the overall distribution of rejections relative to the position in the candidate list, as depicted in figure 5.10, the top-2 segments clearly contain more acceptable keyphrases than the other segments, while in the last segment of the candidate list (90–100%) a sharp decline of acceptable items is observable. This is in line with what has been found in the quantitative evaluation (cf. figure 5.6), where clearly the top of the candidate list accumulates most matches, including partial ones. Investigating the data of the rejections directly, supported by the distribution over the rejection reasons displayed in figure 5.11, as an overall trend, the proportion of rejected items where the reason was given as "too general" is outstanding in all segments of the candidate list. Additionally, rejections classified as "too specific" remain more or less stable and with a rather low impact over the whole list, but a relatively large proportion of "nonsense" rejections is observable in the segment of 0–10% . Therefore, this segment shall be isolated and examined in more detail, with the hope that such an investigation will uncover the reasons for the strong presence of mentioned "nonsense" rejections.

From 61 occurrences of "nonsense" annotations in the first 10% of the candidate list, 26 were due to the text-conversion mistakes and sentence boundary mismatches as pointed out above. This is with 42% a considerably larger proportion than the 28%

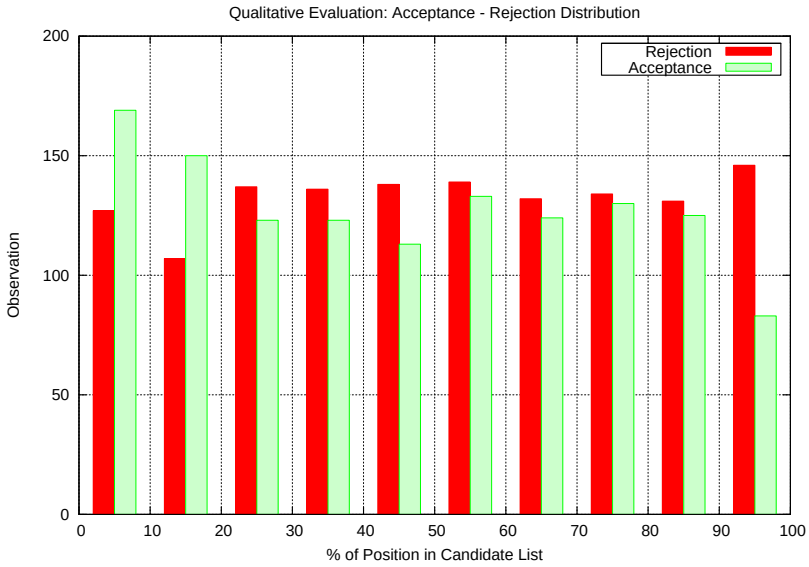

Figure 5.10: Accept vs Reject, Absolute Distribution

overall ratio of nonsense-based errors introduced by such types, as reported above. A number of examples from this segment for sentence boundary mismatches are "Face Recognition Face recognition", "Semantic Data Semantic data", "Scheduling Events Scheduling events" and "speci cation" ↦ `specification`, "bu ers" ↦ `buffers` and "work ows" ↦ `work flows` for text conversion errors. A reduction of these errors would mean that the "nonsense" proportion in this segment drops to a value that more smoothly resembles the behaviour observed in the following three segments for "nonsense" rejections.

Given the results as reported and discussed here, the next section will offer more detailed explanations on a number of observations, and to some extent speculate on actions potentially leading to an improvement of the algorithm.

5.4 Interpretation

Starting with the positive findings, in both experiments it could be observed that the majority of "good" keyphrase predictions was distributed at the beginning of the candidate list. This makes it feasible to implement simplistic measures such as cutting off at a certain position, while still retaining a greater number of acceptable keyphrases. Considering the information obtained by figure 5.6, at around 40% of the candidate list the contribution of "good" candidates starts to fall below significant values. The qualitative evaluation suggests that "bad" candidates begin to gain ground on the

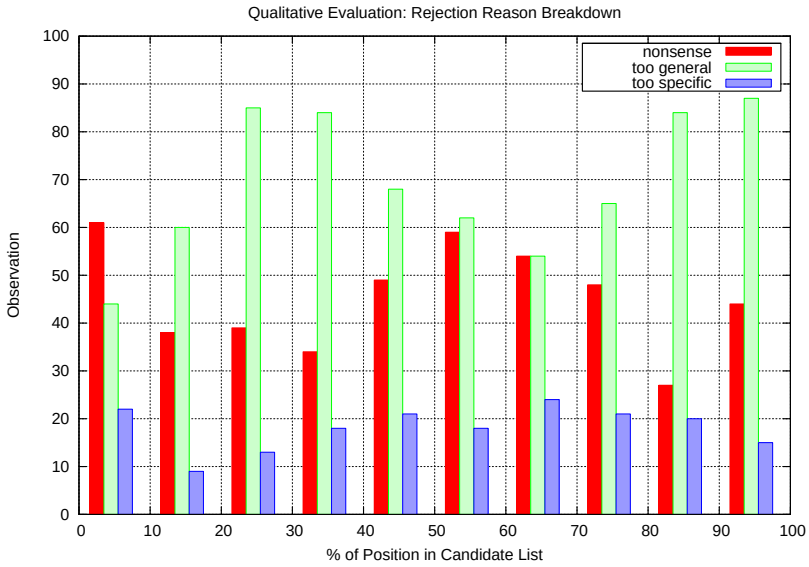

Figure 5.11: Breakdown of Reject Reason, Absolute Distribution

"good" ones after 20% as shown in figure 5.10, however an outnumbering does not take place until after 90% of the candidate list. Thus, a good position for a cut-off will lie somewhere in between 20% and 90%, however in reality it will depend on the practical setting and the luxury to include a number of misses.

As discussed before, one problem for both experiments has been the large number of unmatched / not accepted predictions, leading to a sharp decline in precision. The second experiment also gave further insights into the nature of rejections, and the large proportion of rejections was rated as "too general" to be used as an index term. This may be true when each single one is viewed in isolation, although when the whole set of produced candidates is taken into consideration, it could be argued that the general phrases receive their context from the candidates that have been found acceptable. However, this still does not help to use the general terms for indexing, or further automatic processing such as ontology selection (e.g., as outlined in chapter 2.1.3). About one third of the general candidates were single-word phrases, thus it would be possible to exclude such predictions altogether. Unfortunately, around 43% of all accepted candidates consisted of a single word, which would mean those would also be lost. As in a number of cases, single-word candidates found as too general were already included in keyphrase predictions of larger word-cardinality, a post-processing step could be implemented that discards the proposal of such single-word phrases in case they are already part of another keyphrase.

Barker & Cornacchia reported on using head-noun information when counting frequencies [3]. This information is not used in this approach, as the noun phrase chunker employed here does not provide head-noun annotation off the shelf. However, if head-noun information could be implemented and exploited at the stage of similarity grouping instead of using the Monge-Elkan string metric (cf. section 4.2.3), a larger number of complex terms would be merged into fewer clusters at that level, resulting in a reduced number of candidates, as at most one representative of each cluster is presented as candidate.

A detailed examination of the rejections classified as "nonsense" revealed that a considerable amount resulted from text-conversion errors and mistakes at the very beginning of the linguistic processing pipeline. While it is very difficult to undo mistakes caused by ligature-to-text conversion as a preprocessing step, phrases garbled and broken by hyphenation could be bypassed easily. Moreover, heuristics could be implemented to correct in particular the errors introduced by the sentence boundary mismatches. Most of the time, these errors exhibit a "repetitive pattern", as seen in the examples. Currently, a complex term constructed as a result of the described sentence boundary mismatch is assigned a very high similarity with the other cluster members, which is not diminished by repetitions – furthermore, due to its length, the complex term receives an additional boost, making it likely to be chosen as a representative. A simple string-based test for repetition could be sufficient to reduce this error class in its entirety. If the latter operation would be implemented and executed **before** complex term formation (cf. section 4.2.3), less complex terms would enter the cluster, and the similarity maximisation function which chooses the best representative for the given cluster would not be eluded.

In a gold standard based evaluation, both Turney [75] and Frank et al [27] report average extraction of 1.35–1.46 "good" keyphrases when scaling their output to 5 candidates, and 2.20–2.75 "good" keyphrases when scaling the candidate list to 15 predictions, conducting their experiments on the same dataset. In a separate experiment on rather noisy data, Frank et al [27] however report only 0.8 and 1.7 matching keyphrases per 5 and 15 candidates output. Looking at figure 5.5, with an average of 40 keyphrases extracted per document and relying on the fair evaluation metrics (as defined in section 5.2.2), the algorithm described here reaches the first mark, set by Turney and Frank et al, on average by predicting 10 keyphrases, and the second mark by predicting 25 candidates.

In terms of user acceptance, Turney reports on a proportion of 62% predictions rated as "good", when scaling the output to 7 candidates. When considering only the

first 20% of the candidates (corresponding to roughly 6 predictions) generated in the user experiment by the approach described here, an average acceptance proportion of 58% can be observed.

Despite mentioned flaws, the work described here performs comparable with state-of-the-art keyphrase extraction approaches outlined in section 2.3.2.

At this stage, it is very hard to estimate a projection in terms of improvement on acceptance percentage or recall for a best-case scenario, where the sentence-boundary error class is eliminated completely. Too many variables have to be considered for a projection, and a speculation – whether enthusiastic or not – would not do justice to the effort put into this evaluation, to the anonymous judges who contributed in their free time, and to this approach as a whole. For instance, complex terms introduced as a result of a repair mechanism for sentence boundary mismatches could end up as seperate keyphrase candidates, or not, depending on their status as representative of a cluster. Even if they are proposed as candidate they could be rejected for a number of different reasons. However, with a realistic indication of a good cut-off position, with a number of weaknesses identified (sentence-boundary error type), and with a strategy for improvements (consideration of head nouns, modification of similarity clustering step), it should be obvious that number games are not necessary unless they can provide a clear indication, or truthful estimation, which is not the case here. Thus, it will be interesting to see how the implementation of the discussed potential improvements in another development cycle are reflected in the overall performance of the algorithm.

Chapter 6

Conclusion

This thesis has described the underlying assumptions and the development of a software component capable of extracting keyphrases from textual documents. Next, the contribution will be summarised, along with its limitations, before an outline of future work, which is expected to lead to performance improvement, will conclude the thesis.

6.1 Contribution

The focus of this thesis was the conceptual development, implementation and evaluation of a software component facilitating the identification of keyphrases, which targets the initial creation of free-form metadata from textual documents. Thus, the approach is not concerned with the tasks carried out in classical Information Extraction or Automatic Term Recognition, which nevertheless overlap in some points, and share similarities in the techniques utilised. The underlying considerations have been outlined and the steps undertaken have been described in detail. The resulting software artifact has been implemented in Java as a number of GATE plugins, thereby making extensive use of resources the framework provides off the shelf. Where necessary, the framework has been extended appropriately by additional plugins (language identification, stopword analyser, frequency analyser, etc.), which can also be used independently.

The keyphrase extraction has already been deployed in a number of different scenarios. It has been integrated into the information visualisation workbench for exploratory document collection analysis (IVEA) [1], where it helps to extract prominent phrases

[1] `http://smile.deri.ie/projects/ivea` – retrieved 2008-08-19

that are not yet part of an underlying ontology for semantic annotation. It has been deployed on the KDE-NEPOMUK semantic desktop as part of the semantic note taking tool Semn [2], and as component of the TextAnalytics service [3] in the NEPOMUK-Eclipse Social Semantic Desktop [4]. The component is described online [5], where it is available in a variety of customised packages, including a library API for integration into other projects. A web-interface exposes the functionality for testing and demo purposes, and is accessible from the project page.

It was the aim to produce a knowledge poor solution capable of multilingual text processing – so far, the keyphrase extraction can deal with English, German and French documents. Many of the linguistic resources (stopword and frequency lists) necessary to support additional languages have been put in place, such that the burden of integrating extra languages is lowered to some degree. The approach has been extensively evaluated for English documents, both against a gold standard, where recall was determined at 51.8%, and as a user study, with an average acceptance rate of 49%. Both experiments revealed that a bigger proportion of "good" keyphrases is found at the beginning of the candidate list, confirming the suitability of the confidence ranking function. These initial findings are encouraging and suggest that the algorithm has the potential of outperforming state-of-the-art approaches such as KEA and Extractor/GenEx, given the identified flaws are eliminated and mentioned ideas for improvements are realised.

6.1.1 Limitations

However, not all is well, and a number of problems have been isolated, first and foremost the large amount of not acceptable keyphrase predictions. This poses insofar a problem as users have to choose their preferred keyphrases from an extended set of candidates. It has been shown that this problem is partly caused by the open scenario the application is targeted at: real-world textual data – even on the desktop – is scruffy, and the utilisation of linguistic preprocessors makes the algorithm vulnerable to non-linguistic phenomena, such as unknown characters or words introduced by ligatures or hyphenation, caused by malfunctioning text conversion from PDF documents. The benefit however is a layer of linguistic information that can be well exploited for the task at hand, and as Bourigault observes,

[2] `http://smile.deri.ie/projects/semn` – retrieved 2008-08-19

[3] `http://dev.nepomuk.semanticdesktop.org/wiki/doc/ServiceDescription/TextAnalysis` – retrieved 2008-08-18

[4] `http://nepomuk.semanticdesktop.org/`,`http://dev.nepomuk.semanticdesktop.org/` – retrieved 2008-08-18

[5] `http://smile.deri.ie/projects/keyphrase-extraction` – retrieved 2008-08-18

> *[...] limitations come from the strong hypotheses and the methodological choices which have already been outlined. But in the field of Linguistic Engineering, exceptions do not have the same status as in Linguistic Science; it is here a question of* compromise. [5]

Some of the underlying problems will be revisited as part of further work, and outlined subsequently, which also will conclude this thesis.

6.2 Future Work

The analysis of the data gathered during the evaluation process has yielded valuable leads, which should be followed in order to improve the algorithm in one particular aspect: the reduction of text-conversion errors and sentence-segmentation mistakes. As already described in section 5.4, primarily sentence-segmentation errors could be easily eliminated by scanning for a repetitive pattern at complex term formation. Further, complex terms where de-hyphenation did not succeed, should be bypassed altogether in order to reduce noise which eventually would be classified as "nonsense"-rejection. While such an effort contributes little to produce more acceptable keyphrases, it reduces the amount of "bad" keyphrases generated, which is also most welcome as the user is not confronted with a large number of unacceptable candidates to choose from.

Moreover, as briefly outlined in section 5.4, while the Monge-Elkan string metric does a good job from a cost-benefit perspective, using headnoun information instead would lead to better results at the stage of similarity grouping (cf. section 4.2.3). This however would require the adjustment of the (so far, very simple) noun chunker, but the benefit of one additional exploitable linguistic property makes this path a high priority on the list of things to do next. With clusters containing semantically closer related phrases, the benefit of finding modifiability characteristics could be explored in order to identify the least varying candidate and select it as representative of the cluster (cf. section 4.2.4), similar to the strategies proposed by Wermter & Hahn [76, 77](cf. section 2.3.1). Still, the problem of sparse data needs to be overcome in order to be successful along those lines, as the Wermnter & Hahn approach operates on very large corpora whereas here, only one single document is considered as input. However, while an investigation into such methods could prove helpful to reduce the amount of single-word predictions, it is too early to say which impact it would have on the large proportion of candidates classified as "too general", and therefore purely speculative.

After releasing the keyphrase extraction component to a small number of individuals, the support for more languages other than English, German or French has been

requested, i.e. for Polish. Therefore, an additional path to follow is certainly the extension of multilingual capability, which implies the incubation of additional linguistic processing resources into the GATE framework. Here, mainly part-of-speech tagger and lemmatiser (and to a lesser degree, noun chunker) need to be provided as GATE plugins for further target languages.

Strictly speaking, the adoption of new use-cases for keyphrase extraction is not so much concerned with the approach as such, it is instead a new stage for different, more sophisticated approaches operating on textual data, enabled by reliable prediction of keyphrases, which is now available off the shelf. Therefore, future work will also see further integration of keyphrase extraction with expertise mining [6] and ontology selection as described in section 2.1.3. Beneficial to this work is its rather generic nature, enabling its utilisation in very diverse use-cases. With my hands dirty now it may be time to move on, but keeping these new frontiers in mind, it will be most intriguing to observe how the work described in this thesis can positively contribute to the success of those mentioned application scenarios.

[6] `http://purl.oclc.org/projects/expert` – retrieved 2008-08-18

Appendix A

Third Party Software

Here, third party software libraries are listed which are included in the currently shipped version of the keyphrase extraction component. This does not involve packages that are used with the version deployed as web-interface or web service.

A.1 ngramj

- Languages supported:
  Bulgarian (bg), Czech (cz), Danish (da), German (de), Ewe (ee), Greek (el), English (en), Spanish (es), Estonian (et), Finnish (fi), French (fr), Hungarian (hu), Icelandic (is), Italian (it), Lithuanian (lt), Latvian (lv), Maltese (mt), Dutch (nl), Norwegian (no), Polish (pl), Portugese (pt), Romanian (ro), Russian (ru), Slovak (sk), Slovenian (sl), Swedish (sv), Thai (th), Ukrainian (uk) [1]

- Language profiles generated additionally:
  Irish (ie), Chinese (zh)

- License:
  Lesser GNU Public License `http://www.gnu.org/copyleft/lesser.html` – retrieved 2008-07-14

- Location:
  `http://ngramj.sourceforge.net/` – retrieved 2008-07-13

[1] however, Ukrainian is represented as "ua" internally

A.2 SecondString

- Location `http://secondstring.sourceforge.net/` – retrieved 2008-08-20
- License: University of Illinois/NCSA Open Source License `http://www.otm.uiuc.edu/faculty/forms/opensource.asp` – retrieved 2008-08-20

Appendix B

Lexical Resources

B.1 Stopword Lists

- Languages available:
  Arabic (ar), Bulgarian (bg), Czech (cz), German (de), English (en), Spanish (es), Finnish (fi), French (fr), Hungarian (hu), Italian (it), Polish (pl), Portugese (pt), Romanian (ro), Russian (ru), Slovak (sk)

- Location:
  `http://members.unine.ch/jacques.savoy/clef/index.html` – retrieved 2008-08-20

B.2 Tagsets and Mapping

The English tagset uses a slightly enriched version of the Penn Treebank tagset. The following is the mapping:

B.2.1 English

```
CC
CD --> DETERMINER_OR_PRONOUN
DT --> DETERMINER_OR_PRONOUN
EX
FW
IN
JJ --> ADJECTIVE
JJR --> ADJECTIVE
JJS --> ADJECTIVE
JJSS --> ADJECTIVE
-LRB-
LS
MD --> VERBAUXMOD
NN --> NOUN
NNP --> NOUN
NNPS --> NOUN
NNS --> NOUN
NP --> NOUN
NPS --> NOUN
PDT
POS
PP --> DETERMINER_OR_PRONOUN
PRPR$ --> DETERMINER_OR_PRONOUN
PRP --> DETERMINER_OR_PRONOUN
PRP$ --> DETERMINER_OR_PRONOUN
RB --> ADVERB
RBR --> ADVERB
RBS --> ADVERB
RP
STAART
SYM
TO
UH
VBD --> VERB
VBG --> VERB
VBN --> VERB
```

```
VBP --> VERB
VB --> VERB
VBZ --> VERB
WDT
WP$
WP
WRB
::
,
$
-
-
-
(
.
#
)
-
```

B.2.2 French

The tagset for French tags is shipped with the TreeTagger and without further documentation. The following is the mapping:

```
ABR
ADJ --> ADJECTIVE
ADV --> ADVERB
DET:ART --> DETERMINER_OR_PRONOUN
DET:POS --> DETERMINER_OR_PRONOUN
INT
KON
NAM --> NOUN
NOM --> NOUN
NUM --> DETERMINER_OR_PRONOUN
PRO --> DETERMINER_OR_PRONOUN
PRO:DEM --> DETERMINER_OR_PRONOUN
PRO:IND --> DETERMINER_OR_PRONOUN
PRO:PER
PRO:POS --> DETERMINER_OR_PRONOUN
PRO:REL
PRP
PRP:det
PUN
PUN:cit
```

```
SENT
SYM
VER:cond --> VERB
VER:futu --> VERB
VER:impe --> VERB
VER:impf --> VERB
VER:infi --> VERB
VER:pper --> VERB
VER:ppre --> VERB
VER:pres --> VERB
VER:simp --> VERB
VER:subi --> VERB
VER:subp --> VERB
```

B.2.3 German

The Stuttgart-Tübingen Tagset (STTS) is used for German part-of-speech tags. The following is the mapping:

```
ADJA --> ADJECTIVE
ADJD  --> ADJECTIVE
ADV  --> ADVERB
APPR
APPRART
APPO
APZR
ART --> DETERMINER_OR_PRONOUN
CARD --> DETERMINER_OR_PRONOUN
FM
ITJ
KOUI
KOUS
KON
KOKOM
NN --> NOUN
NE --> NOUN
PDS --> DETERMINER_OR_PRONOUN
PDAT --> DETERMINER_OR_PRONOUN
PIS
PIAT --> DETERMINER_OR_PRONOUN
PIDAT --> DETERMINER_OR_PRONOUN
PPER
PPOSS
PPOSAT --> DETERMINER_OR_PRONOUN
```

```
PRELS
PRELAT
PRF
PWS
PWAT --> DETERMINER_OR_PRONOUN
PWAV
PAV
PTKZU
PTKNEG
PTKVZ
PTKANT
PTKA
TRUNC
VVFIN --> VERB
VVIMP --> VERB
VVINF --> VERB
VVIZU --> VERB
VVPP --> VERB
VAFIN --> VERB
VAIMP --> VERBAUXMOD
VAINF --> VERBAUXMOD
VAPP --> VERBAUXMOD
VMFIN --> VERBAUXMOD
VMINF --> VERBAUXMOD
VMPP --> VERBAUXMOD
XY
$,
$.
$(
```

B.3 JAPE Grammar for Noun Chunking

The JAPE grammar which is used to implement noun chunking as GATE plugin:

```
/*************************************************************
 * JAPE Grammar for a simple Noun Chunker.
 * This approach is actually a 1-to-1 adapation of the chunking
 * strategy described in the LingPipe [1] part-of-speech tutorial,
 * where a chunking implementation is also covered briefly [2].
 *
 * The Java-code described as mentioned above is simply translated
 * into a bunch of JAPE macros and rules, for NounChunks.
 *
 * Furthermore, the part-of-speech tags of the Brown-Corpus had
 * to be adjusted/mapped to the tagset used by the GATE part-of-speech
 * tagger resource.
 *
 * [1] http://alias-i.com/lingpipe/index.html
 * [2] http://alias-i.com/lingpipe/demos/tutorial/posTags/read-me.html
 *
 * author: Alexander Schutz (alex.schutz@deri.ie)
 * last change: 2008-10-14
 *************************************************************/

Phase: NounChunker
Input: Token Phrase
Options: control=appelt

// Nouns:
// NN, NN$, NNS, NNS$, NP, NP$, NPS, NPS$
//  ---> POS-Tags of Hepple Tag Set
Macro: NOUN
(
 {Token.category == "NN"}
 | {Token.category == "NNP"}
 | {Token.category == "NNPS"}
 | {Token.category == "NNS"}
 | {Token.category == "NP"}
 | {Token.category == "NPS"}
)

// Adverbs:
// RB, RB$, RBR, RBT, RN (not RP, the particle adverb)
//  ---> POS-Tags of Hepple Tag Set
```

```
Macro: ADVERB
(
 {Token.category == "RB"}
 | {Token.category == "RBR"}
 | {Token.category == "RBS"}
)

// Determiners & Numerals:
// ABN, ABX, AP, AP$, AT, CD, CD$, DT, DT$, DTI, DTS, DTX, OD
//  ---> POS-Tags of Hepple Tag Set
Macro: DETERMINER
(
 {Token.category == "CD"}
 | {Token.category == "DT"}
 | {Token.category == "PDT"}
)

// Adjectives:
// JJ,JJ$,JJR,JJS,JJT
//  ---> POS-Tags of Hepple Tag Set
Macro: ADJECTIVE
(
 {Token.category == "JJ"}
 | {Token.category == "JJR"}
 | {Token.category == "JJS"}
 | {Token.category == "JJSS"}
)

// Pronoun:
// PN,PN$,PP$,PP$$,PPL,PPLS,PPO,PPS,PPSS
//  ---> POS-Tags of Hepple Tag Set
Macro: PRONOUN
(
 {Token.category == "PP"}
 | {Token.category == "PRP"}
 | {Token.category == "PRP$"}
 | {Token.category == "PRPR$"}
)

// Punctuation:
// ', ``, '', ., (, ), *, --, :, ,
//  ---> POS-Tags of Hepple Tag Set
Macro: PUNCT
```

```
(
 {Token.category == ","}
 | {Token.category == "."}
 | {Token.category == "’"}
 | {Token.category == "("}
 | {Token.category == ")"}
 | {Token.category == "::"}
 | {Token.category == "-"}
 | {Token.category == "#"}
)

Macro: NOUNCHUNK_START
(
 DETERMINER | ADJECTIVE | NOUN | PRONOUN
)

Macro: NOUNCHUNK_CONT
(
 NOUNCHUNK_START | ADVERB
)

Rule: NounChunk1
Priority: 2
(
 (NOUNCHUNK_START)
 (
  (NOUNCHUNK_CONT)
 )*
):nounchunkref
-->
{
  gate.FeatureMap features = Factory.newFeatureMap();

  gate.AnnotationSet nounchunk = (gate.AnnotationSet)bindings.get("nounchunkref");
  gate.Annotation nounChunkAnn = (gate.Annotation) nounchunk.iterator().next();
  features.put("rule", "NounChunk1");

  annotations.add(nounchunk.firstNode(), nounchunk.lastNode(), "NounChunk", features);
}
```

B.4 Multilingual Frequency Lists from Internet Corpora

- Languages available:
  German (de), English (en), Spanish (es), Finnish (fi), French (fr), Italian (it), Japanese (jp), Portugese (pt), Russian (ru), Chinese (zh)
- Location:
  `http://corpus.leeds.ac.uk/list.html` – retrieved 2008-08-20

German (de) :

- corpus size: 187,789,449
- lexicon size: 3,932,190
- list elements: 5,000

English (en) :

- corpus size: 181,376,006
- lexicon size: 1,701,333
- list elements: 5,000

Spanish (es) :

- corpus size: 143,567,378
- lexicon size: 1,579,383
- list elements: 15,000

Finnish (fi) :

- corpus size: 168,256,557
- lexicon size: 2,656,018
- list elements: 5,000

French (fr) :

- corpus size: 185,102,375
- lexicon size: 1,476,288

- list elements: 5,000

Italian (it) :

- corpus size: 12,113,422
- lexicon size: 261,566
- list elements: 10,000

Japanese (jp) :

- corpus size: 253,071,774
- lexicon size: 124,489
- list elements: 15,000

Portugese (pt) :

- corpus size: 193,321,224
- lexicon size: 2,170,013
- list elements: 5,000

Russian (ru) :

- corpus size: 156,534,391
- lexicon size: 791,311
- list elements: 5,000

Chinese (zh) :

- corpus size: 281,660,631
- lexicon size: 1,268,440
- list elements: 15,000

Appendix C

Evaluation Form

C.1 Instructions

Email Instructions

The following instructions were given to the anonymous, self-selected judges, per email, sent to various mailing lists:

1. Point your browser to the URL
   `http://srvgal66.deri.ie:8080/keyword-evaluation-webapp`

2. Enter a nickname - no worries, it will be encrypted, so not even I can see what name you originally entered. This is used for associating evaluations with each other in case you liked it so much that you want to do another one (yes yes yes please!).

3. Pick a document you know **very** well, i.e. that you are (co-)author of

   - Documents should be in English for now, we can try different languages at another time
   - Documents can be of type PDF, HTML, PlainText. Sometimes even Microsoft Word works but that is a bit of a gamble.
   - For optimal results, the document should have reasonable size. Whatever you find on you desktop, your favourite paper of the life sciences, the rejection you got because none of the reviewers really understood what you were doing, or that essay you wrote back in the days about Georgie Best's magical dribblings which is ok not to have heard about or one of the BBC

news stories. Just don't pick books of the size of the Holy Bible, or ANSI Common Lisp, or very short documents consisting of very few words.

As a rule of thumb, anything in between 500 and 50,000 words shall be fine.

4. Upload the document you have settled on via the web interface. Alternatively, you may also point to a URL.

5. Once you submit, the extraction process starts and it may take a little while, depending on the size of the document. It should be really fast for short documents (<=5,000 words) though.

6. You are presented with an interface which enables you to **accept** or **reject** the predictions. Every prediction is marked as accept by default, so please take a look at each one and give your nod. In case of rejection, please select the (somewhat) more detailed options which will be enabled only then (**too general**, *too specific*, *nonsense*). The instructions on the website will also give examples as a guideline.

   Too general : for instance, a predicted keyphrase time refers to a very generic concept, which could be applied in almost any context. Therefore, the reason for rejection would be too general

   Too specific : at the other end of the scale, if, for instance, a document talks about digital cameras, in particular about model IUXS, and on one occasion makes a reference to another model Superphoto, then if the latter is predicted as keyphrase, it is obviously too specific.

   Nonsense : the third option, nonsense, should be used if a prediction does not seem to make sense, either because the words are ill-formed (cut-off at hyphenation, suspiciously long, etc.), mostly resulting from filtering errors in the underlying process.

7. Once you are finished, hit the submit button again, your results will be transmitted. and you have offered a big service to me.

Appendix D

Glossary

ANNIE	A Nearly New Information Extraction system
API	Application Programming Interface
ATE	Automatic Term Extraction
ATR	Automatic Term Recognition
BNC	British National Corpus
CLAWS	Constituent Likelihood Automatic Word-tagging System
FSA	Finite State Automata
GATE	General Architecture for Text Engineering
GNU	GNU's Not Unix
HMM	Hidden Markov Model
IE	Information Extraction
IR	Information Retrieval
LT	Language Technology
MEDLINE	MEDlars onLINE
MUC	Message Understanding Conference
NAO	NEPOMUK Annotation Ontology

NER	Named Entity Recognition
NIE	NEPOMUK Information Elements
NLP	Natural Language Processing
NRL	NEPOMUK Representation Language
OWL	Web Ontology Language
PoS	part-of-speech
P2P	peer-to-peer
PIMO	Personal Information Model
PP	Prepositional Phrase
RDF	Resource Description Framework
RDFS	RDF Schema
SOA	Service Oriented Architecture
TF/IDF	Term Frequency / Inverse Document Freqency
TnT	Text and Trigrams
UMLS	Unified Medical Language System

Bibliography

[1] Sophia Ananiadou and Hideki Mima. An Application and Evaluation of the C/NC-value Approach for the Automatic term Recognition of Multi-Word units in Japanese. *International Journal of Terminology*, 6(2):175–194, 2000.

[2] Eric Atwell, John Hughes, and Clive Souter. AMALGAM: : Automatic Mapping Among Lexico-Grammatical Annotation Models. In J Klavans, editor, *The Balancing Act: Combining Symbolic and Statistical Approaches to Language - Proceedings of the ACL Workshop*, pages 21–28. Association for Computational Linguistics, 1994.

[3] Ken Barker and Nadia Cornacchia. Using Noun Phrase Heads to Extract Document Keyphrases. In *AI '00: Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence*, pages 40–52, London, UK, 2000. Springer.

[4] Kenneth R. Beesley. Language Identifier: A Computer Program for Automatic Natural-Language Identification of On-line Text. In *Proceedings of the 29th Annual Conference of the American Translators Association*, pages 47–54, October 1988.

[5] Didier Bourigault. Surface grammatical analysis for the extraction of terminological noun phrases. In *Proceedings of the 14th conference on Computational linguistics*, pages 977–981. Association for Computational Linguistics, 1992.

[6] Thorsten Brants. TnT – A Statistical Part-of-Speech Tagger. In *Proceedings of the 6th Applied Natural Language Processing (ANLP)*, pages 224–231, Seattle, WA, 2000.

[7] Christopher Brewster, Harith Alani, Srinandan Sasmahapatra, and Yorick Wilks. Data Driven Ontology Evaluation. In *Proceedings of International Conference on Language Resources and Evaluation (LREC)*, pages 24–30, 2004.

[8] Eric Brill. A Simple Rule-Based Part of Speech Tagger. In *Proceedings of the third conference on Applied natural language processing (ANLP)*, pages 152–155, Trento, Italy, 1992. Association for Computational Linguistics.

[9] Paul Buitelaar, Thomas Eigner, and Thierry Declerck. OntoSelect: A Dynamic Ontology Library with Support for Ontology Selection. In *Proceedings of the Demo Session at the International Semantic Web Conference, Hiroshima, Japan*, November 2004.

[10] Jean Carletta. Assessing Agreement on Classification Tasks: the Kappa Statistic. *Computational Linguistics*, 22(2):249–254, 1996.

[11] Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named Graphs, Provenance and Trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM.

[12] Paul-Alexandru Chirita, Stefania Costache, Siegfried Handschuh, and Wolfgang Nejdl. P-TAG: Large Scale Automatic Generation of Personalized Annotation TAGs for the Web. In *Proceedings of the 16th international conference on World Wide Web*, pages 845–854, Banff, Alberta, Canada, 2007. ACM.

[13] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[14] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In Subbarao Kambhampati and Craig A. Knoblock, editors, *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03)*, pages 73–78, Acapulco, Mexico, August 2003.

[15] Nigel Collier, Chikashi Nobata, and Junichi Tsujii. Automatic acquisition and classification of terminology using a tagged corpus in the molecular biology domain. *Journal of Terminology, John Benjamins*, 7(2):239–258, 2001.

[16] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[17] Béatrice Daille, Éric Gaussier, and Jean-Marc Langé. Towards Automatic Extraction of Monolingual and Bilingual Terminology. In *Proceedings of the 15th conference on Computational linguistics COLING*, volume 1, pages 515–521, Kyoto, Japan, 1994. Association for Computational Linguistics.

[18] Fred J. Damerau. Generating and evaluating domain-oriented multi-word terms from texts. *Information Processing and Management*, 29(4):433–447, 1993.

[19] Mathieu d'Aquin, Marta Sabou, Martin Dzbor, Claudio Baldassarre, Laurian Gridinoc, Sofia Angeletou, and Enrico Motta. Watson: A Gateway for Next Generation Semantic Web Applications. In *Proceedings Poster Session of the International Semantic Web Conference (ISWC2007), Busan, Korea.*, Busan, Korea, 2007.

[20] Stefan Decker and Martin Frank. The Social Semantic Desktop. Technical Report 2, Digital Enterprise Research Institute, May 2004.

[21] Stefan Decker, Sergey Melnik, Frank Van Harmelen, Dieter Fensel, Michel Klein, Jeen Broekstra, Michael Erdmann, and Ian Horrocks. The Semantic Web: The Roles of XML and RDF. *IEEE Internet Computing*, 04(5):63–74, September 2000.

[22] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal C Doshi, and Joel Sachs. Swoogle: A Search and Metadata Engine for the Semantic Web. In *CIKM '04: Proceedings of the 13th ACM Conference on Information and Knowledge Management*, pages 652–659, Washington, D.C., USA, November 2004. ACM Press.

[23] Ted Dunning. Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

[24] Ronen Feldman, Moshe Fresko, Yakkov Kinar, Yehuda Lindell, Orly Liphstat, Martin Rajman, Yonatan Schler, and Oren Zamir. *Text Mining at the Term Level*, volume 1510/1998 of *Lecture Notes in Computer Science*, chapter 8, pages 65–73. Springer-Verlag, 1998.

[25] Jenny Finkel, Shipra Dingare, Huy Nguyen, Malvina Nissim, Christopher Manning, and Gail Sinclair. Exploiting Context for Biomedical Entity Recognition: From Syntax to the Web. In Nigel Collier, Patrick Ruch, and Adeline Nazarenko,

editors, *COLING 2004 International Joint workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP) 2004*, pages 91–94, Geneva, Switzerland, August 2004. COLING.

[26] Christopher Fox. A stop list for general text. *SIGIR Forum*, 24:19–21, 1990.

[27] Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. Domain-specific keyphrase extraction. In *Proceedings 16th International Joint Conference on Artificial Intelligence*, pages 668–673, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

[28] Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. Automatic Recognition of Multi-Word Terms: the C-value/NC-value Method. *International Journal on Digital Libraries*, 3(2):117–132, 2000.

[29] Katerina T. Frantzi, Sophia Ananiadou, and Junichi Tsujii. The C-value/NC-value Method of Automatic Recognition for Multi-word Terms. In *Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries*, volume 1513 of *Lecture Notes in Computer Science*, pages 585–604. Springer Verlag, 1998.

[30] Gregory Grefenstette. Comparing Two Language Identification Schemes. In *Proceedings of 3rd International Conference on Statistical Analysis of Textual Data (JADT 95)*, pages 263–268, Rome, Italy, December 1995.

[31] Tudor Groza, Siegfried Handschuh, Knud Moeller, Gunnar Grimnes, Leo Sauermann, Enrico Minack, Cedric Mesnage, Mehdi Jazayeri, Gerald Reif, and Rosa Gudjonsdottir. The NEPOMUK Project - On the way to the Social Semantic Desktop. In Tassilo Pellegrini and Sebastian Schaffert, editors, *Proceedings of I-Semantics 2007*, pages 201–211. JUCS, 2007.

[32] Tudor Groza, Siegfried Handschuh, Knud Möller, and Stefan Decker. SALT - Semantically Annotated LaTeX for scientific publications. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *The Semantic Web: Research and Applications, 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3-7, 2007, Proceedings*, volume 4519 of *Lecture Notes in Computer Science*, pages 518–532. Springer, 2007.

[33] Tudor Groza, Leo Sauermann, and Paul-Alexandru Chirita. NEPOMUK Deliverable D6.1: Semantic based knowledge systems - First version Backbone and Connector Infrastructure. technical report, NUIG, DFKI, L3S, November 2006.

[34] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[35] Mark Hepple. Independence and commitment: Assumptions for Rapid Training and Execution of Rule-based POS Taggers. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL-2000)*, pages 278–285, Hong Kong, 2000. Association for Computational Linguistics.

[36] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading Massachusetts, 1 edition, 1979.

[37] Steve Jones, Stephen Lundy, and Gordon W. Paynter. Interactive Document Summarisation Using Automatically Extracted Keyphrases. In *Proceedings of the 35th Hawaii International Conference on System Sciences*, volume 4, pages 101–111, Hawaii, 2002. IEEE Computer Society.

[38] Steve Jones and Gordon W. Paynter. Automatic extraction of document keyphrases for use in digital libraries: Evaluation and applications. *Journal of the American Society for Information Science and Technology*, 53(8):653–677, April 2002.

[39] Adam Kilgarriff. Which words are particularly characteristic of a text? a survey of statistical approaches. In *Proceedings AISB Workshop on Language Engineering for Document Analysis and Recognition*, pages 33–40, Sussex, United Kingdom, April 1996.

[40] Adam Kilgarriff. Language is never, ever, ever, random. *Corpus Linguistics and Linguistic Theory*, 1(2):263–276, November 2005.

[41] Adam Kilgarriff and Tony Rose. Measures for corpus similarity and homogeneity. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing (EMNLP-3)*, pages 46–52, Granada, Spain, 1998.

[42] Geoffrey Leech, Roger Garside, and Michael Bryant. CLAWS4: The tagging of the British National Corpus. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING 1994)*, pages 622–628, Kyoto, Japan, 1994.

[43] Vanessa Lopez, Victoria Uren, Enrico Motta, and Michele Pasin. AquaLog: An ontology-driven question answering system for organizational semantic intranets. *Journal of Web Semantics*, 5(2):72–105, 2007.

[44] Julie Beth Lovins. Development of a Stemming Algorithm. *Mechanical translation and computational linguistics*, 11:22–31, 1968.

[45] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[46] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.

[47] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[48] Yutaka Matsuo and Mitsuru Ishizuka. Keyword Extraction From A Single Document Using Word Co-Occurrence Statistical Information. *International Journal on Artificial Intelligence Tools*, 13(1):157–169, 2004.

[49] Diana Maynard and Sophia Ananiadou. Identifying Terms by their Family and Friends. In *Proceedings of the 18th conference on Computational linguistics*, pages 530–536, Saarbrücken, Germany, 2000. Association for Computational Linguistics.

[50] Diana Maynard and Sophia Ananiadou. TRUCKS: a model for automatic term recognition. *Journal of Natural Language Processing*, 8(1):101–125, December 2000.

[51] George A. Miller and Edwin B. Newman. Tests of a Statistical Explanation of the Rank-Frequency Relation for Words in Written English. *The American Journal of Psychology*, 71(1):209–218, March 1958.

[52] Knud Möller, Siegfried Handschuh, Sebastian Trüg, Laura Josan, and Stefan Decker. Demo: Visual Programming for the Semantic Desktop with Konduit. In *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, volume 5021 of *Lecture Notes in Computer Science*, pages 849–853. Springer, 2008.

[53] Alvaro E. Monge and Chales Elkan. The Field Matching Problem: Algorithms and Applications. In *Proceedings of Knowledge Discovery and Data Mining*, pages 267–270, 1996.

[54] Ted Pedersen, Mehmet Kayaalp, and Rebecca Bruce. Significant Lexical Relationships. In Howard Shrobe and Ted Senator, editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, volume 2, pages 455–460, Portland, OR, August 1996. AAAI Press.

[55] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. Towards a Theory of Natural Language Interfaces to Databases. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157, Miami, Florida, USA, January 2003. ACM.

[56] Borislav Popov, Atanas Kiryakov, Damyan Ognyanoff, Dimitar Manov, and Angel Kirilov. Kim – a semantic platform for information extraction and retrieval. *Natural Language Engineering*, 10(3-4):375–392, 2004.

[57] Martin F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.

[58] Lance A. Ramshaw and Mitchell P. Marcus. Text Chunking using Transformation-Based Learning. In David Yarovsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Somerset, New Jersey, 1995. Association for Computational Linguistics.

[59] Paul Rayson and Roger Garside. Comparing Corpora using Frequency Profiling. In *Proceedings of the workshop on Comparing corpora*, pages 1–6. Association for Computational Linguistics, 2000.

[60] Gerald Reif, Tudor Groza, Siegfried Handschuh, Mehdi Jazayeri, and Cedric Mesnage. NEPOMUK Architecture Deliverable D6.2.A: Semantic based knowledge systems Intermediate. Technical report, DFKI, NUIG, USI, University of Zürich, September 2007.

[61] Beatrice Santorini. Part-of-Speech Tagging Guidelines for the Penn Treebank Project. Technical report, Department of Computer and Information Science, University of Pennsylvania, March 1991.

[62] Leo Sauermann, Ansgar Bernardi, and Andreas Dengel. Overview and Outlook on the Semantic Desktop. In Stefan Decker, Jack Park, Dennis Quan, and Leo Sauermann, editors, *Proceedings of the 1st Workshop on The Semantic Desktop at the ISWC 2005 Conference*, volume 175 of *CEUR Workshop Proceedings*, pages 1–19. CEUR-WS, November 2005.

[63] Jacques Savoy. A Stemming Procedure and Stopword List for General French Corpora. *Journal of the American Society for Information Science*, 50(10):944–952, July 1999.

[64] Jacques Savoy. Combining Multiple Strategies for Effective Monolingual and Cross-Language Retrieval. *Information Retrieval*, 7:121–148, 2004.

[65] Anne Schiller, Simone Teufel, and Christine Thielen. Guidelines fuer das Tagging deutscher Textkorpora mit STTS. Technical report, Institut für Maschinelle Sprachverarbeitung, University Stuttgart, 1995.

[66] Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK, 1994.

[67] Francesco Sclano and Paola Velardi. TermExtractor: a Web Application to Learn the Common Terminology of Interest Groups and Research Communities. In *Proceedings of the 7th Conference Terminologie et Intelligence Artificielle (TIA'2007)*, Sophia Antipolis, France, October 2007.

[68] Claude E. Shannon. A Mathematical Theory of Communication. *Bell Telephone System Technical Publications*, 27:379–423, 623–656, 1948.

[69] Michael Sintek, Ludger van Elst, Simon Scerri, and Siegfried Handschuh. Distributed Knowledge Representation on the Social Semantic Desktop: Named Graphs, Views and Roles in NRL. In *The Semantic Web: Research and Applications. 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3-7, 2007. Proceedings*, volume 4519 of *Lecture Notes in Computer Science*, pages 594–608. Springer, 2007.

[70] Lorraine Tanabe, Natalie Xie, Lynne H. Thom, Wayne Matten, and W. John Wilbur. GENETAG: a tagged corpus for gene/protein named entity recognition. *BMC Bioinformatics*, 6:S3, 2005.

[71] Simone Teufel. A Support Tool for Tagset Mapping. In *Proceedings of SIGDAT. Workshop in cooperation with EACL 95*, Dublin, 1995. European Chapter of the Association for Computational Linguistics.

[72] VinhTuan Thai, Siegfried Handschuh, and Stefan Decker. Tight Coupling of Personal Interests with Multi-dimensional Visualization for Exploration and Analysis of Text Collections. In *Information Visualisation, 2008. IV '08. 12th International Conference*, pages 221–226, July 2008.

[73] Tzong-Han H. Tsai, Shih-Hung H. Wu, Wen-Chi C. Chou, Yu-Chun C. Lin, Ding He, Jieh Hsiang, Ting-Yi Y. Sung, and Wen-Lian L. Hsu. Various criteria in the evaluation of biomedical named entity recognition. *BMC Bioinformatics*, 7(1):92, February 2006.

[74] Giovanni Tummarello, Christian Morbidoni, and Michele Nucci. Enabling Semantic Web Communities with DBin: An Overview. In *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings (Semantic Web Challenge Track)*, volume 4273 of *Lecture Notes in Computer Science*, pages 943–950. Springer, 2006.

[75] Peter D. Turney. Learning Algorithms for Keyphrase Extraction. *Information Retrieval*, 2(4):303–336, 2000.

[76] Joachim Wermter and Udo Hahn. Collocation extraction based on modifiability statistics. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, pages 980–986, Geneva, Switzerland, 2004. Association for Computational Linguistics.

[77] Joachim Wermter and Udo Hahn. Paradigmatic Modifiability Statistics for the Extraction of Complex Multi-Word Terms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, page 843–850, Vancouver, October 2005. Association for Computational Linguistics.

[78] Joachim Wermter and Udo Hahn. You can't beat frequency (unless you use linguistic knowledge) – a qualitative evaluation of association measures for collocation and term extraction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, page 785–792, Sydney, July 2006. Association for Computational Linguistics.

[79] George Kingsley Zipf. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology.* Addison-Wesley Press Inc., Cambridge 42, MA, USA, 1949.