Sandesh: Response Management System

Pankaj Jangid, Ripul H. Gupta, M. Sasikumar
ETU division, NCST
RainTree Marg, Sector 7, Belapur
Navi Mumbai 400 614, Maharashtra, India
Email: {*pankaj, ripul, sasi*}*@ncst.ernet.in*

Abstract

E-mail is the primary mode of communication in most online learning scenarios. The faculty gets overloaded with this mode. So there is a need for a system that can reduce the load of query responding on the faculty. *Automated e-mail responding systems* have been in existence for a long time. But they are capable of responding only to some selected keywords in the subject line or they send the same reply to all the mails. An example is the vacation program often used when somebody is on vacation. What we need is an *intelligent system* that can understand the content of an e-mail, and try to find a response for it; if a suitable response is not found the mail may be redirected to concerned person. The query mail and the answer produced should be remembered by the system so that a similar query next time need not go to the faculty, but will be handled by the system itself. In this paper, we describe *Sandesh*, a system developed based on this approach.

1 Introduction

Even in the traditional instruction model driven by face to face lectures, interaction plays a significant role for clearing doubts, reinforcing concepts, seeking clarification, etc. A good part of learning also takes place through this. Such interactions become still more important in an e-learning scenario, where often the medium of instruction is passive web content. Let us first look at the interaction scenario in a traditional classroom environment.

In the traditional education system students generally ask queries in class. Every person present in the class benefits from the queries asked by others and the responses to these. Also faculty has to answer the query only once or twice. At any rate, the number of times a faculty answers a query does not grow with the number of students in a class. Some students, however, may be absent for the class or may be slow in understanding the subject. Absent students might ask the same questions later and weak students might want a more detailed answer. The problem here is that the faculty has to repeat

the answer or she might have to elaborate with some repetition for weak students, which could be a waste of time for other students. This is also very cumbersome to the instructor.

Some students feel shy to ask questions in front of other students. The shy student may ask the instructor her question separately later or may rely on self-study for clearing her doubts. This means that other students do not get to know of such questions and they may be faced with a similar doubt later on. The students that are not part of the discussion are, therefore, not benefited and may lose out on some useful points.

The teacher might want to refer to something before answering some of the doubts; she might need some time for the same and hence likely to forget about the question. In such a case, if the students don't ask the question again they might never get the answer.

Weak students may also suffer due to fast learners. For example, if a bright student asks a question which requires a deep understanding of the concept being discussed, the teacher may go into intricate explanations, which the weak students find difficult to grasp. It may happen that most of the class is lost due to the discussion involving two, the teacher and the bright student.

Many of these problems can be addressed effectively when questions are asked through the Internet (e-mail or web). Faculty can explain or elaborate on a particular problem for a particular student. Each student gets the response to her question at a level of detail that the teacher has selected for her. Students who want detailed answers will get that without wasting others time. Now shy students are more likely to ask queries, since there is no intimidating atmosphere of standing in front of a number of people, her questions and answers being heard by others, etc. However the burden on the instructor has only increased. The instructor here would have to repeat her responses many times to different students who may ask the same question in different ways. The time and effort required to understand and respond to queries increase since more students ask questions. This may translate to significant delays in response to queries.

One problem in asking queries via the Internet is that she might not get answer to her query immediately. The faculty might check her mails after a long time or she may choose to refer some books before replying.

The answer to above problem could be to use a *discussion board*. The primary advantage of the *discussion board* over e-mail is that quite a few of the queries will be answered by students themselves. But it also carries some of the negative points of the classroom such as

- A shy student might hesitate to post since the questions and their responses are visible to others.
- The students posting the question might not take the effort of going through previously answered posts and might ask the same question again.
- Also, as students themselves might answer some questions, the control on the discussion board may become weak.

The last point can be addressed by using a *moderated discussion board*. But it again burdens the faculty as she has to not only approve but also check every message that has been posted.

Administrative Queries: Educational institutes generally receive a lot of queries related to admissions, fees, qualifications, etc. particularly during admission time. Someone has to manually check the queries and reply regularly. Quite often there will be many queries from different sources, phrased differently, coming at different times but addressing the same issue, and requiring the same answer. There are many problems which are clearly visible in such scenarios. Much of the information requested by such queries about the institution may already be on their web page, handbook, FAQs, etc. Queries relating to such information can be best answered by reproducing two or three relevant paragraphs from the relevant sources. It may not be essential to tailor the response to each query individually. For issues not covered adequately by these sources, a human expert may need to formulate a reply. These responses, can then be added to the pool expanding the coverage of the information pool. Thus a response management system based on this approach can be of significant help for administrative query handling also.

Section 2 briefly discusses the existing systems and approaches to this problem. We examine the issues in developing an automatic response management system (RMS) keeping in mind these aspects in section 3. Section 4 describe *Sandesh*, the system we are implementing. Section 5 discusses the current status and further work ahead.

2 Related Work

A few systems are available which try to reduce the burden of responding to the queries. Basically these systems manage pre-formulated responses, and suggest answers using them.

Visnet MailFlow [VM] is an e-mail tracing system. Depending upon the query received, the system forwards the e-mail to the concerned person. And it will also check whether a response has been sent to the client. It also maintains threads. Email Manager [EM] is also a similar system. RightNow [RN] replies to e-mails with links to answers. Maintains threads of communication. ReplyMate [RM] is a plug-in for MS Outlook Express. It manages a pool of replies. It suggests some replies for a query, from which a user can select one and send.

Companies, which have already started offering online courses, hire extra support people called mentors for this work. But in times like these, when it is becoming more and more difficult to hold on to even the faculty, holding on to these support persons is difficult. Training for such support persons also takes quite a lot of effort from the faculty. Besides, hiring mentors adds to the cost.

It is necessary to reduce the effort of the faculty spent on responding to queries without adding the overhead of training a support person every time a new one is hired.

Case based reasoning(CBR) [Kolodner, 1993] is a potential technology relevant to this problem. CBR systems use previous instances to formulate response to new systems. The concept of using earlier query-response mails as a resource for responding to fresh queries has a similar structure. However, CBR systems focus heavily on organised case structure to facilitate matching and indexing of cases. As of now, we are assuming most queries to be unstructured free-text queries and hence CBR model is not being pursued.

3 Automatically Responding to Queries

The problems listed above can be eliminated to a large degree, by building a system that makes use of existing information and responds to queries automatically. Today, we have a large number of technologies including sophisticated natural language processing (NLP) tools, information retrieval models and web technologies, which can be exploited to address this task satisfactorily.

The information or data needed to respond to a query can be obtained from any of the following sources:

Website: Information may be available on the *organisation web-site* in the form of course details, FAQs, registration details, resources, course schedule, module results, etc.
Local Resources: An educational institute would have course and module handbooks whereas a company might have manuals containing product details. These *local resources* can be used to search for information.
Previous Replies: Since a majority of the queries are of a similar nature, *earlier replies* to e-mails may be utilised for answering repetitive queries.
The Faculty: Specific queries not answered before, may have to be answered by *the faculty* (or *the domain expert*).

While replying to a query we need to decide from where to pick relevant information among these sources. The choice can be based on the following criteria:

- Probability of finding the data.
- Reliability of data.
- Cost of using the resource.

Out of the four resources listed above, the fourth choice is the most reliable but also the most expensive. The next in reliability is the previously responded queries, but initially the data might not be present; so one needs to build up a collection before it really starts being useful. Next in line are the manuals and handbooks. This, in a way, is potentially the most reliable data. However its utility reduces due to the fact that the information passed would be fixed, and may or may not answer the query directly. For clarification, we still might have to go to the faculty. The least reliable source of information is the web. A lot of data is present on the web, but finding relevant data from the web can be very difficult. And because of low reliability, the responses sent have to be monitored. Thus we see that relying on any one of the data resource might not be sufficient.

Broadly, the approach to query response can be outlined as follows:

- Subject line analyser, reads the subject line and if a valid command is recognised, replies from the pool of predefined responses to the commands.
- Compare the query against a database of previously answered queries; if an adequate match is found, use the response to the matching query.

- If the new query does not match any of the previous queries, the system should consult the manuals and handbooks to find the relevant information. But the information in these books might not be properly formatted to be sent as a reply, so the reply is first forwarded to the faculty for any corrections. Such validation may not be required in all domains. Hence, this can be kept as a configurable option.
- If the local resources do not contain the relevant data, the system should try to find it on the web. And the response thus formulated should be forwarded to the faculty for approval.
- If all of the above tries fail, the mail should be sent to the faculty to be replied.
- Whatever mails are replied by the faculty, should pass again through the system, so that the system could extract the reply for use later for a similar query.

4 Sandesh Approach

Sandesh follows the outline sketched in the previous section. This section discusses some of the major implementation aspects in realising the various operations in the outline algorithm. The issues discussed are representation, classification and processing of query.

4.1 Modelling the Raw E-mail

Apart from the matching of query (contained in the mail) we need to extract various other information from the e-mail. For example, we need the subject field to check if it is a pre-formatted template query. We need the sender field for validation of user, to construct response, etc. Thus the first step in our approach is to capture the mail in a convenient internal format.

```
<QUERY id="">
<FROM name="" email=""></FROM>
<REPLYTO name="" email=""></REPLYTO>
<CC>
  <TO name="" email=""></TO>
  <TO name="" email=""></TO>
</CC>
<SUBJECT>Subject</SUBJECT>
<BODY>Body</BODY>
<SIGNATURE>Signature</SIGNATURE>
</QUERY>
```

Figure 1: Query after XML conversion.

Received e-mails are converted into XML format [Abiteboul et al, 2000]. We chose XML because it allows us to create custom tags which are suitable for our understanding. See figure 1 for the XML form of a query. Note that the headers have been converted to more readable form. In this form it is easy to debug and analyse the system if some error occurs. Also various modules can use this data. If the standard e-mail format changes in future then we have to change only the module which converts e-mail into XML. This will enable us to handle multiple query formats later on, without burdening other modules with handling various formats.

4.2 E-mail Classification

The e-mail received by *Sandesh* could be any of these types:

- Fixed Command mails,
- Reply of a mail (from a faculty),
- Follow-up Query (from a student),
- Free-text mails.

If the subject of the e-mail contains a valid command that the system understands, the e-mail is assumed to be of the fixed command type. The e-mail could also be a faculty's response for a query. *Sandesh* adds these e-mails into its previous query database, so that they can be used later. If the e-mail is a followup e-mail from a student, then while processing it we have to take into consideration the previous e-mail. If the e-mail does not fit any of the above categories, it is classified as a free-text mail and the body of the mail is used for processing.

Thus we need to find out the type of mail that we have received before we can process it.

Usually when an e-mail is sent, a "Message-ID" is generated by the mail client, which is of the form `unique_string@host.domain` [RFC-821]. The host domain is taken to make the "Message-ID" unique. When we send a reply to an e-mail an "In-Reply-To" header [RFC-822] is attached. The value of this header is the "Message-ID" of the mail to which we are replying. Using this header we can determine whether it is a follow up e-mail or a faculty response or a new query. Further an e-mail can be classified as a fixed command e-mail if the subject line contains a valid command. If any of the above cases is not applicable then it is a free-text e-mail.

4.3 E-mail Processing

After categorising the mails we need to process them according to the category they belong to. The rest of this section explains the types of processing done depending upon the e-mail category.

4.3.1 Fixed Command Mails

For these types of e-mails, we have some predefined commands which are defined in an XML configuration file [Abiteboul et al, 2000]. These commands consist of some combination of key words not necessarily in order and a few allowed variables. e.g *cst result 20020217001* can be used to fetch the CST examination result of student with id 20020017001.

The configuration file has two parts: definition part and command set part. The definition part contains data type definitions. The command set part contains commands which *Sandesh* would recognise.

```
<DEFINITIONS>
  <ROLLNO match="[[:alpha:]][[:digit:]]{7}"/>
  <CSTROLLNO match="[[:digit:]]{11}"/>
  <CSTEVENT match="(result|exam(ination)?)"/>
  <MODULE match="(oopj|dsal|pccp|coos|dbms|cnet)"/>
  <COURSE match="(pgdst|apgdst|fpgdst|pgdit)"/>
  <TEST match="(quiz|mgpt|demo)"/>
  <TIME match="(time|date)"/>
</DEFINITIONS>
```

Figure 2: Definition part of configuration file.

Figure 2 describes that ROLLNO type contains one alphabet followed by 7 digits, CSTROLLNO is an eleven digit entity and so on. Similarly other data types are defined using POSIX's extended regular expressions [Friedl, 1997].

```
<COMMAND>
  <WORD value="CST"/>
  <WORD value="RESULT"/>
  <VAR value="CSTROLLNO"/>
  <REPLY>
    <URL method="POST" value="http://202.141.151.1:9898/CstRollno">
      <FORMVAR name="stuid" value="CSTROLLNO"/>
    </URL>
  </REPLY>
</COMMAND>
```

Figure 3: A portion of command configuration file which shows a command.

A command is composed of several key words and variables. Figure 3 describes a command, which contains words "cst" and "result" along with a variable of type "CSTROLLNO" defined in the definition section, figure 2, of the configuration file. The command will match only if the subject contains at least the three components: two specified words and a variable of type "CSTROLLNO".

When a mail is received it is first compared with all the commands to find a match. If a match is found the mail is assumed to be a *fixed command e-mail*. The action required for this can be obtained from the "REPLY" sub-tag of the "COMMAND" tag. This action could vary from sending a predefined answer to retrieving some data from the web/database or even making new entries to the database or sending an e-mail. The variable in the command is used to obtain the dynamic result. For example, variable "CSTROLLNO" can be used to query for the result of a particular students. The data received as a result of the query will be sent back as the reply. Similarly a fixed command can be used by a student to register for a particular module, in this case the action part will be "to insert a record into the database". Fixed command can also be used by the sender to bypass the automatic processing of the e-mail and forward it directly to the faculty.

4.3.2 Free-Text E-mails

When the query is identified to be a *free-text e-mail*, it is converted as a stream of words, which is then passed through a query enricher. This enriched query is sent to the module, which searches in the cache of previous e-mails. If no satisfactory response is found, then the handbooks will be searched followed by the web. When a satisfactory result is obtained, the reply is sent.

Query enricher: The Query Enricher removes all the stop words like articles and prepositions. All remaining words are considered to be important and are stemmed. Duplicate words are removed. It also fetches equivalent words from thesaurus; this helps to handle synonyms in a domain specific manner. These steps improve the effectiveness of the matching stages to follow.

Previous responses: The enriched query will be compared with the enriched queries of the previously replied mails. If the percentage of similarity between both the enriched queries goes above a configured threshold, then a suitable match is supposed to be found. The answer will be the previously asked question and the reply for it. Depending upon the mode set in the configuration the reply will be either sent directly, or will be forwarded to the faculty for her consent. See figure 4 for an example of auto-reply from cache of previous responses.

```
Query:
What is the period of validity for CST scores ?

Reply:
This is an automated reply. Reply is selected from replies to
earlier queries. Earlier query with reply:

How long are the CST scores valid ?

The CST scores are valid for 41 months, for consideration for
improvement, recruitment to NCST and admission to courses run by
NCST and its affiliates. Their validity for consideration of
admission to MCA/M.Tech/DIIT programmes depend on the decisions of
the institutions concerned. The CST score report has an unlimited
validity period for other use.
```

Figure 4: Example of reply from cache of previous responses.

Local Resource and Web: The handbook and web are used next to find some matching text. Here again the enriched query is matched with the index words of these local resources. Each document has index words associated with it. Index words are extracted by *Sandesh* by finding important words from the document [Rijsbergen, 1985]. These keywords are used to find the degree of relevance of a document for a particular query. This is done for all documents and the top n document with the highest relevance measure are used. n is a configurable measure.

A whole document might be very long to be used as a reply to a query. We first need to segregate the documents into small chunks. In *Sandesh*, we divide a document into paragraphs. In case of a text document we assume a blank line to be a paragraph break. In case of an HTML document, header

tags, paragraph tag or two consecutive line breaks are assumed to be paragraph boundaries. After selecting top n relevant documents, as explained earlier, paragraphs of these documents are matched with the enriched query. Degree of matching is the number of words that are common to both the paragraph and the enriched query. If the degree of matching is above a threshold the paragraph can be included in the reply. The reply is sent directly to the sender or to the faculty for approval as per the configuration settings.

Mail Routing: Once the system has decided that the e-mail has to be forwarded to the faculty, it also needs to decide the person to whom the e-mail has to be forwarded. This can be achieved by having a few keywords for each of the faculty. The system matches the incoming query with these keywords. The query is then forwarded to the best matched e-mail address.

4.3.3 Followup Mails

Some fixed command e-mails might trigger a process, which requires some verification before the processing is done. For example, a student applying for registration for a module. Before the student is actually registered, we need to find out if the mail was genuine. For this case, the system can send an e-mail to the e-mail address of the student and when confirmation is received the original e-mail is assumed to be authentic. Another case of a followup e-mail could be when the sender is not satisfied with the response to her previous e-mail. In such a case, the system instead of trying to find reply for it, should forward it to the faculty. For these purposes, we need to maintain the state of the e-mails. A followup e-mail can be recognised using the e-mail headers (e-mail id), as explained earlier.

4.3.4 Faculty Response

When a mail is received by the system and if it is not able to confidently reply to the mail, it will forward the mail to the faculty. When response is received from the faculty, the system must forward the mail to the original sender and store the answer to the query for future use.

Some queries and responses may be so specific that it has little reuse value. In such a case, the system should not cache the response. *Sandesh* provides an annotation for the faculty to indicate that the e-mail should not be cached.

5 Current Status and Future Work

Currently *Sandesh* has an extensible auto-responding subject line mode. It uses regular expressions to define new commands. Besides pre-formulated replies *sandesh* can produce replies for new queries from a given corpus.

Also, *Sandesh* can be configured for one domain at a time. If one wishes to use sandesh for multiple domains then she would need to use multiple e-mail addresses. To tackle this problem, we would need to do the categorisation of the documents as well as of the queries.

All the configuration files are in XML format and currently it supports the following major configuration parameters:

Log level: How much information you want to write into the log file, for tracing and debugging.
Disable: (*Module name*) to disable one module out of the subject parsing, cache search or free-text modules.
Threshold: The degree of matching that can be assumed to be good match.
Faculty e-mail Addresses: The e-mail addresses of the faculty and the associated keywords.

The current system when replying from previous mails does not take care of the semantics, emotions, etc. For example if a person asks, "when to do something", *Sandesh* may answer for, "when not to do something". So negation is not taken care of. It does only statistical match. So we need to store the semantics of such words as "not" in this case. Only IR techniques [Rijsbergen, 1985] are being used till now to find the match between any resource and the data sources, but in future we intend to use more of *Natural Language Techniques* (NLT) to understand the query and the resources better. Use of NLT will also help us in determining the emotional content of the message.

Current version of *Sandesh* runs on an Pentium class processors with 128MB RAM and 5MB of space. The system is being developed for POSIX compliant platforms. It works on systems which support the UNIX *mail* and *procmail* utilities.

Acknowledgement

This work was carried out under the Vidyakash project of the Department of Information Technology, Ministry of Communications and Information Technology, Government of India. We also thank the referees for their comments and suggestions on this paper.

References

[Abiteboul et al, 2000] Serge Abiteboul, Peter Buneman and Dan Suciu. *Data on the web: Relations to semistructured data and xml*. Morgan Kaufman Publishers, 2000.

[EM] Email Manager. Response manager. http:/www.ifmodules.com/main.php/email_manager.

[Friedl, 1997] Jeffrey E F Friedl. *Mastering regular expressions: Powerful techniques for perl and other tools*. Oreilly and Associates, 1997.

[Kolodner, 1993] Janet Kolodner. *Case Based Reasoning*. Morgan Kaufmann, 1993.

[RFC-821] Postel J. B. Simple Mail Transfer Protocol. ftp://ftp.rfc-editor.org/in-notes/rfc821.txt

[RFC-822] Crocker, D. H. Standard for the Format of ARPA Internet Text Messages. ftp://ftp.rfc-editor.org/in-notes/rfc822.txt.

[Rijsbergen, 1985] C. J. Van Rijsbergen. *Information Retrieval*. Butterworths Scientific Ltd, 1985.

[RM] ReplyMate. MS Outlook Express Plug-in. http:/www.replymate.com/.

[RN] RightNow. Email responder. http:/www.rightnow.com/products/email.html.

[VM] Visinet MailFlow. Response manager and e-mail tracking system. http:/www.deerfield.com/products/visnetic_mailflow/.