

Sandesh – An Automatic Email Response System

Aparna R., Sasikumar M.
Educational Technology Division,
Centre for Development of Advanced Computing,
Kharghar, Navi Mumbai-400614
India
{aparna, sisi}@cdacmumbai.in

Abstract - E-learning has become very popular in the last few years and more and more people are switching to this mode of education. The primary mode of communication between the faculty and the students, in an e-learning scenario, is email. This mode of communication requires the faculty to answer the same query a number of times and this increases the burden on the faculty. We propose an email response system called Sandesh which addresses this issue. Sandesh uses NLP techniques to identify previous query-responses which match the new query and forwards the same response to the student. It also caters to fixed format queries. Sandesh also allows students to bypass it and directly contact the faculty. Further enhancements to support web and other local resources to generate responses are in progress.

I. BACKGROUND

There has been a tremendous growth in information and communication technologies in the last few years. Computers are fast entering every aspect of our life. Right from banking, income-tax, railway reservation, aircraft reservation, to even education, computers are leaving no fields to capture. When we talk about education, we can see the popularity that e-learning has gained in the educational circuits. Advances in the hardware technology and the steady reduction in hardware cost have made it possible for many institutions to afford the online mode to impart knowledge.

E-learning is bringing significant amount of changes in the educational models. This includes shifting from a teacher-centric model of education to a learner-centric model of education. The 'anytime-anywhere' paradigm of education has become popular with e-learning. This style of learning enables the learner to learn the concepts at his pace. These style changes are becoming acceptable to people. One major area of concern in e-learning is the teacher-student communication.

In a traditional classroom lecture model, the faculty and the students interact face to face when the faculty delivers his lecture. Here, whatever the faculty speaks every student gets to hear. Similarly, if a student asks a query to the faculty, every student hears the query as well as the faculty's response to it. Hence, the probability of the same query being asked by someone else is very low. The only case of repetition, here, may be when a student is absent for a lecture. Since the student was not present during the session, the teacher may get queries which may have already been discussed during the session and the faculty would have to repeat his explanation for the same.

Some of the drawbacks of the traditional model are:

- Given that a weak students' grasping capacity is low, they will need more elaborate explanation of concepts

which may not be of interest to the other students.

- It may happen that a bright student asks a query and the faculty responds to it with a 'to-the-point' answer. But this may not be useful for a weak student with the same query as he may find the explanation inadequate. In such a case, the student may choose to just keep quite and as a result his query will never get resolved.
- Sometimes the faculty may need some time before answering to a certain query as he or she may want to look up some material before answering. Most of the time this is not possible in the typical classroom lecture mode. Even if the faculty tells the student that he would answer the question later, it is possible that he may forget about it and the student may not remind him about the same.
- Shy students may not ask their queries in the class. Such students may choose to ask their queries to the faculty after the lecture, in which case the other students are not benefited by the same, or may choose to not ask the question at all, in which case their doubts never get cleared. In any case, shy students never ask questions during the first few lectures of their course - hence their basics remain unclear often.

Many of these drawbacks are non-existent in the online learning model where the primary mode of communication between the students and the faculty is via emails or discussion boards. Email encourages shy students to come out of their shells and ask their queries. Also, since this mode of communication is one to one, the faculty can frame his response depending on whether the student is a bright student or a weak student. Thus, a response to a weak student can be elaborate while a response to a bright student can be precise. This mode also gives the faculty the needed time to refer to other materials to answer queries.

One of the biggest issues with this model of interaction is that the number of times a faculty may potentially need to answer a query increases with the number of students. This happens because email being between just two people, others are not aware of the communication. This is unlike a traditional learning environment, where the faculty and students communicate face to face, as mentioned earlier. Though there maybe times when a faculty repeats the answer (e.g. when the student was absent in the class), the number of times the faculty needs to do so, does not grow with the number of students.

Hence, unlike in a normal classroom lecture where every

student can hear the communication between the faculty and any other student, in an e-learning mode, when a faculty responds to a particular query the remaining students are not benefited by the same. As a result the faculty has to answer to the same query several times. This can burden the faculty tremendously.

The other issues with email communication are:

- Since emails are not a completely reliable mode of communication, they can be lost without information. Thus, if a student's mail is lost, he will all the time keep thinking that the faculty has not answered to his query, whereas the faculty would not be aware of any such mail. Similarly, if the faculty sends a response to the student and it gets lost, the faculty will not be aware of it.
- It is possible that the faculty is on leave but the students are not aware of this and send him mails. Hence the faculty is unable to reply to the queries but the students only get a picture that the faculty is not responding.
- It is also possible that the faculty is busy with some other work when he receives the mail, though he does manage to read the query and decides to respond to it at a later time. But then he forgets this later and as a result the query remains unanswered.

One can argue that one can create email groups, so that everyone can see what question a fellow student has asked and hence the faculty may not have to answer the same query again and again.. But again, when a student posts a query he may not necessarily check all the earlier mails in the group to see if the query had been asked earlier or not. And all the other above mentioned problems still remain. To add to this, email groups may cause shy students to get into their shells. Hence email groups offer no significant benefits, except for sharing.

In this paper, we propose an automated email response system (Sandesh) which addresses many of these concerns. The next section introduces the idea and gives an outline of such a system. Section 3 looks at relevant solution and related work. Section 4 discusses Sandesh's approach in detail. In Section 5 we talk about the current status of Sandesh. Section 6 concludes the paper. Section 7 talks about the future work and enhancements.

II. AUTOMATED E-MAIL RESPONSE SYSTEMS

Most of the above discussed problems is reduced to a large extent by having a system to automatically respond to queries via emails. The students' mails will be intercepted by this system which checks if a response to the query is already available. If yes, then it forwards the response to the student. Thus the faculty is freed from the burden of replying to the same query over and over again. Automatic Email response systems are feasible today, thanks to sophisticated Natural Language Processing techniques and tools, information retrieval models and web technologies. Automatic Email Response systems can also integrate other sources in the interaction circuit. These include:

- Web: A lot of material will be available on the

institution's website which can be made use of to answer queries.

- Local resource: Institutions will have course handbooks which will have sections like Frequently Asked Questions (FAQs) which will answer a lot of queries.
- Query Response Repository (QRR): These systems generally maintain a repository of previously answered queries. The repository is updated regularly with new queries and responses.
- Faculty: If no response is obtained from the above mentioned resources then the system contacts the faculty to get the response.

Of the above listed resources, the faculty is the most reliable but the most expensive resource. The next in line is Query Response Repository. The system can improve on the quality of response retrieval by making use of good natural language processing methods. The next in line of reliability are the Local Resources. The only problem with this resource is that we may not find the exact response to a query. The last reliable resource of the four is the web. There is a lot of data on the web but finding relevant data is not easy. Hence the response from this resource needs to be monitored.

This can be combined with the email support for fixed format queries. A number of common queries from students would be for exam results, exam dates, lecture dates, fee schedule, etc. These can be supported by allowing emails with a fixed format subject line like 'Results *StudentID ExamID*' which can be processed easily and a reply mail generated by looking up the appropriate data source. Our system allows for such queries also, by filtering such queries first and processing them separately. Fixed format queries are important since they provide high reliability in response, but they are very restrictive in their scope.

The broad approach of an automatic email response system, to obtain a response to a query, is as follows:

1. Analyse the subject line to see if it matches a fixed format query. Mails with some fixed subject patterns have standard solutions to generate responses.
2. If the subject line is not a fixed type, then check if a response to the query is already available in a repository of old responses (QRR). If an adequate match is found, then forward the response to the student.
3. If no response is found, check the local resources like online handbooks, to find a suitable response and if obtained forward the same to the student.
4. If no response is obtained, then forward the mail to the faculty.
5. All responses from the faculty should again pass through the system so that it can store the response in its repository for future use.

III. SYSTEMS CURRENTLY AVAILABLE AND RELATED WORK

There are a few systems that attempt to reduce the burden of responding to queries. These systems manage pre-formatted

responses to respond to queries, or decide whom to forward the mail to or suggest responses to queries.

Visnet MailFlow [9] is an email tracing system. It decides whom to forward the query to, depending upon the query received. It also keeps track of whether the response to the mail has been sent or not. EmailManager [2] is another system on similar lines. RightNow [7] replies to emails with links to answers. ReplyMate[6] is a plug-in for MS Outlook Express which manages pool of replies and suggests responses to queries from the pool.

IV. OUR APPROACH

We propose an automated mail response system - SANDESH - that will act as a middleman between the faculty and the students. All the queries posed by the students to the faculty is intercepted by Sandesh, which checks its repository to see if the same or similar query has been asked earlier and if a response to it is already available. If yes, then Sandesh forwards the corresponding response to the student; else it forwards the mail to the faculty. The response from the faculty is also intercepted by Sandesh so that it can update its repository. Once it has updated the repository with the faculty's response, it forwards the mail to the concerned student. Sandesh also keeps track of whether a pending query has been answered by the faculty or not.

Thus, the burden on the faculty is reduced considerably, for he does not have to respond to the same query several times, thanks to Sandesh which presents the faculty mostly with unique queries for answering. This is not the only advantage. The question answer interaction often encompasses a lot of specialised knowledge. The growing repository offers a simple and effective device to capture these. The interaction and associated knowledge fragments become accessible to all learners at their own pace - when they need it.

The broad procedure that is followed by Sandesh to perform the above is given below:

1. Check if the mail is a fresh mail, reply from faculty or a follow-up mail. A follow-up mail is a mail from a student indicating he is not happy with the auto generated response received earlier, and want to talk to the faculty directly.
2. If the mail is a fresh mail then check if it is fixed format mail or any other mail.
3. If the mail is a fixed format mail, then execute the corresponding command and forward the result to the student.
4. If the mail is not a fixed format mail, but is a fresh mail, then perform the following sub-tasks:
 - Search the Query Response Repository for a suitable response. If found, then forward the same to the student with a warning that this is an automatically generated response.
 - If a suitable response is not found in the QRR then, identify the faculty who will be able to handle the query. Forward the mail to him and mark the mail request as pending.
5. If the mail is a reply from a faculty then perform the

following tasks:

- Check if the mail is a reply to a follow-up mail or reply to a fresh mail.
 - If it is a reply to a follow-up mail, then remove the corresponding query-mail from the pending list and forward the response to the student.
 - If the mail is a reply to a fresh mail, then perform the following tasks:
 - Remove the corresponding mail from the pending list.
 - Update the Query Response Repository with this query and the faculty's response to it.
 - Forward the faculty's response to the student.
6. If the mail is a follow-up mail then mark it pending and forward the same to the respective faculty.

Currently Sandesh does not include search of web and local resources, since we are focusing on use of previous responses.

Implementing such a system requires addressing a number of design issues. The major ones are discussed below.

A. How to identify similar queries

This is the heart of Sandesh as this will decide how well the system works. Determining similar queries will require Sandesh to check how many useful words of a new mail are present in any other mail. We say 'useful' words because there is no point in counting occurrences of articles/prepositions/conjunctions. We need to understand the domain and find which words would be useful for matching. For example, in an e-learning domain words like 'exam', 'subject', 'course', etc. are important. This matching of ONLY the useful words will improve the result quality. Similarly, a match cannot be one-to-one, i.e. a word 'when' in a query could be matched against a date or time in some other mail. Consider the following example

* When will the MCA entrance exam be conducted this year?

* I heard that this year the MCA exam will be conducted on 13th March. Is this true?

Both the queries above are essentially the same. But there is no 'When' in the second query and no '13th March' in the first. But both these seek the same information and hence should be matched for effective output. Thus a plain word to word match will not do. We will need to consider synonyms and other related words based on the domain for matching.

Another issue in matching is converting all words to their root form. For example, the words 'actively', 'active', etc. should all become 'active'.

B. Relevance of the response

The above process could give us some best response to the query. Simply choosing a best response is not always advisable. There may not be 'adequately' matching queries observed earlier; these are cases where the query should be sent to faculty for answer. The system needs to detect such cases. For this purpose, we set a threshold for every match in the QRR. This threshold, among others, depends on the mail

size. It depends on the number of 'useful words' in the query-response set. If this number ≤ 4 then the threshold is set to 75%. For everything else, the threshold is set to 25%.

C. Bypassing the system

In general, the student may not always be satisfied with the automatically generated response and may want to ask the same query directly to the faculty. In this case he will need to bypass the system. In Sandesh, we generate a subject for every automatic response and request the students to reply to the mail without modifying the subject line in case they want to bypass the system.

D. Identifying the faculty to whom the response needs to be forwarded

In general, a course may involve multiple faculties. Sandesh needs to be able to figure out which faculty is most appropriate to answer a query. A variant of the matching algorithm used for finding similar queries could be used here, since this can be viewed as a mail classification problem. Currently, Sandesh assumes a single faculty account and does not address this problem.

V. SANDESH AT WORK:

At its current stage, Sandesh intercepts a query and performs the following actions:

a) It first checks the subject of the query and compares it with its command formats. If ascertained as a fixed format mail, then Sandesh performs the defined action for the command. The action to be performed can vary from sending a standard response to fetching some dynamic content from the web, like getting a module result of a student.

The Command Structures for Fixed Command mails are defined in a Command File, which is an XML file. We use XML structure because it is one of the popular and standard ways of representing data. Also, XML allows us to have custom-defined tags to represent the data. There are several 'commands' defined in the Command File.

A command is composed of several keywords and variables. The variables in a command are used to retrieve dynamic content. For every command, an action to be performed is defined. The action can vary from sending a predefined answer to retrieving data from the web. If a query has a subject similar to any of the command structures, then the corresponding action is performed and the result is forwarded to the student.

All command structures have two parts – a 'Definition' part as seen in 'Fig. 1' and a 'Command' part as seen in 'Fig. 2'. The 'Command' part defines the structure a command can take, i.e. it defines the variables and the keywords that constitute the command. For example, 'Fig. 2' defines a command with the structure *MODULE RESULT ROLLNO*, where *MODULE* and *ROLLNO* are the defined variables and *RESULT* is a keyword. This command will match a subject line, only if the subject line contains the three specified components: the word *RESULT*

and two other words, one mapping to a *ROLLNO* and one mapping to a *MODULE*.

The 'Command' part also defines the action to be performed in case the subject of any query matches this structure. The *REPLY* tag, seen in 'Fig. 2', defines this. In this case, information is retrieved from the web using the values of *MODULE* and *ROLLNO*. The data retrieved thus, will be sent to the student as response.

What words can map to a variable in a command is defined in the 'Definition' part of the command. For example, 'Fig. 1' defines a variable *ROLLNO* as a structure containing one alphabet followed by 7 digits. Thus, if a subject line has a word starting with an alphabet followed by 7-digits then, it could be identified as a value of *ROLLNO*. Similarly, the variable *MODULE* can have a value 'java' or 'c' or 'webprogramming' or 'networks'. Any of these words occurring in the subject line could be identified as a *MODULE*. No other words will correspond to a *MODULE*. A command can have many such variables defined for it. The variables are defined using POSIX's regular expressions.

```
<DEFINITIONS>
  <ROLLNO match=" [[: alpha:]]{7}" />
  <MODULE
MATCH="(java|c|webprogramming|networks)" />
  <TIME match="(time|date)" />
</DEFINITIONS>
```

Fig. 1 - Definition part

```
<COMMAND>
  <VAR value="MODULE">
  <WORD value="RESULT">
  <VAR value="ROLLNO">
  <REPLY>
    <URL method="POST" value="http://202.141.152.1/Rollno">
    <FORMVAR name="moduleid" value="MODULE"/>
    <FORMVAR name="rollno" value="ROLLNO"/>
  </URL>
  </REPLY>
</COMMAND>
```

Fig. 2 - Command part

Thus, for the command structure defined in 'Fig. 1' and 'Fig. 2', a subject line of the form *d025700 RESULTS java* will match. Here, 'd0257009' is the *ROLLNO* and 'java' is the *MODULE*.

b) If the mail is not a Fixed Format mail, then Sandesh checks its Query Response Repository (QRR) to see if a similar query has been asked before. The QRR is also an XML

file, which is a repository of previously asked queries and responses to them.

An example query-response pair stored in the QRR is shown in 'Fig. 3'.

```

<REPLY>
  <WORD val="FINAL"/>
  <WORD val="YEAR"/>
  <WORD val="STUDENT"/>
  <WORD val="WHICH"/>
  <WORD val="CST"/>
  <WORD val="LEVELS"/>
  <WORD val="ELIGIBLE"/>
  <QUESTION>
    <![CDATA[ I am a final year student. Which levels
of CST am I eligible for?]]>
  </QUESTION>
  <MIN-MATCH val="25"/>
  <BODY>
    <![CDATA[Students must be in the final
year of the respective degree/diploma in order to be
eligible, except for students of MCA, who can write the E,
I or D-level examination. Post-graduate students may apply
for any level for which they are eligible.]]>
  </BODY>
</REPLY>

```

Fig. 3 - QRR snippet

The part inside the QUESTION tag indicates the question and the part inside the BODY tag indicates the response (of some faculty) to the question. The WORD tag indicates the important words for the response. So whenever a new query comes, it is broken into useful words and those words are matched against the WORD tag. If the number of matches is above the threshold limit, defined in the MIN-MATCH tag, then the response is considered suitable and the same is forwarded to the student.

c) If no suitable response is obtained from the QRR then Sandesh forwards the query to the faculty after adding it to its Pending list. 'Fig. 4' shows the structure of the Pending list

```

<PENDING>
  <ID val="NSKJPTYRIZJTNMNGSUMJJHKZHG"/>
  <ADDRESS val="APARNA@NCST.ERNET.IN"/>
  <SUBJECT val="Path"/>
  <QUESTION>
    <![CDATA[In my Graduation I have done BCA (Bachelor
of Computer Applications ) from Indore Univ & Post
Graduation in Business Administration ( PGDBA ) from ITM
Kharghar with sp. in Marketing & IT. Currently I am
associated with TechNova Imaging Systems on Profile of IT
Infrastructure Management. I hereby want to know: Am I
eligible for the above mention course in Project Mgt, Course
Time Table and Fees & Payment Mode]]>
  </QUESTION>
</PENDING>

```

Fig. 4 - Pending List snippet

This list will help Sandesh to send automatic reminders to the faculty.

VI. CONCLUSION

The core part of Sandesh is complete and is planned to be used for our internal requirement first. We have evaluated Sandesh's response in terms of accuracy using data from queries encountered in our diploma programmes. Overall we find the response adequate. However, further test are in progress with a larger corpus.

As a part of future work, we plan to implement the following features in the Sandesh:

- Searching for responses on the web from a predefined list of sites.
- Managing multiple sets of responses as a way to address handling students with different learning pace.
- Providing facility to the users to update the QRR and Command files through the system.
- Automatic identification of faculty with the help of keywords and configuration files. Currently Sandesh assumes a standard faculty account.
- Handling emails with attachments. For mails with fixed subject lines we may want to store the attachments of the mails in a certain directory.

REFERENCES

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

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

[3] Jeffery E F Friedl. *Mastering Regular Expressions: Powerful techniques for Perl and other tools*. O'Reilly and Associates, 1997.

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

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

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

[7] RightNow. Email responder. <http://www.rightnow.com/products/email.html>

[8] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to natural Language Processing, Computational Linguistics and Speech Recognition*.

[9] Visnet MailFlow. Response Manager and Email tracking system. http://www.deerfield.com/products/visnetic_mailflow/

[10] XML and DOM reference. <http://www.w3.org>