ELSEVIER

Data & Knowledge Engineering 38 (2001) 85–100

DATA & KNOWLEDGE ENGINEERING

www.elsevier.com/locate/datak

Using information extraction and natural language generation to answer e-mail ☆

Leila Kosseim *, Stéphane Beauregard, Guy Lapalme

RALI, DIRO, Université de Montréal CP 6128, Succ. Centre Ville, Montréal, Que., Canada H3C 3J7

Received 27 March 2001; received in revised form 27 March 2001; accepted 27 March 2001

Abstract

This paper discusses the use of information extraction (IE) and natural language generation (NLG) in the design of an automated e-mail answering system. We analyse short free-form texts and generate a customised and linguistically motivated answer to frequently asked questions. We describe the approach and the design of a system currently being developed to answer e-mail in French regarding printer-related questions addressed to the technical support staff of our computer science department. A domain-dependent approach was preferred because precision is crucial in e-mail answering, as a wrong answer sent to a client can have negative consequences in terms of customer satisfaction.

Keywords: Information extraction; Natural language generation; Automatic e-mail response

1. Introduction

The number of free-form electronic documents available and needing to be processed has reached a level that makes the automatic manipulation of natural language a necessity. Manual manipulation is both time-consuming and expensive, making natural language processing (NLP) techniques very attractive. E-mail messages make up a large portion of the free-form documents available today and as e-mail becomes more and more popular, an automated e-mail answering service will become as necessary as an automated telephone service is today.

This paper discusses the use of information extraction (IE) and natural language generation (NLG) to answer e-mail automatically. We describe the design of a system currently being developed to answer e-mail in French regarding printer-related questions addressed to the technical support staff of our computer science department. The original project was prompted by a local

☆ A preliminary version of this paper appears in: Proceedings of the 5th International Conference on Applications of Natural Language to Information Systems (NLDB'2000), Versailles, France, June 2000.

* Corresponding author.

E-mail addresses: kosseim@iro.umontreal.ca (L. Kosseim), beaurs@iro.umontreal.ca (S. Beauregard), lapalme@iro.umontreal.ca (G. Lapalme).

0169-023X/01/$ - see front matter
PII: S0169-023X(01)00018-0

corporation for its customer service needs, but because of difficulties in gathering a corpus of e-mail messages from their archives, local e-mails from our department were used for the research.

Unlike typical question answering (QA) systems (e.g. [25,26]) our focus is on *analysing* short, free-form texts and *generating* a customised and linguistically motivated answer to frequently asked questions. In our view two main approaches are available to answer e-mail: (1) information retrieval or (2) information extraction and text generation. With the information retrieval (IR) approach, the incoming message is considered as a query to be matched against some textual knowledge base [e.g., a list of frequently asked questions (FAQ)]. E-mail answering thus becomes a question of finding passages from the textual knowledge base that best relate to the incoming message and sending the passages as is to the user. Although this approach has the major advantage of being domain independent, [1] it does not provide a natural and customised answer necessary for customer service. In addition, this technique provides an ergonomically awkward interaction with e-mail users, and it supposes that the e-mail message is short enough so that the process can be computationally efficient. In order to provide a specific response to the user query, we believe that key information from the content of the e-mail must be identified and used to customise the answer. For this reason, whenever the specific discourse domain of the question is known (e.g., through classification), IE seems in our view more adequate for analysing the incoming e-mail and template-based NLG appropriate to produce a natural answer from pre-written response templates.

2. The corpus

The system we are developing is aimed at answering printer-related user e-mail received by the technical support staff of our department, where communications are in French. We have concentrated our efforts on a specific discourse domain in order to obtain good results in IE and to be able to manage the knowledge representation. The entire corpus covers a 3-year period and is composed of 191 e-mails. The corpus was split into two sets, the first 2 years for analysis (122 messages) and the last year for testing (69 messages). From the original corpus, we kept only the messages that discussed only one topic and were self-contained, i.e., that do not need information external to the message to be understood. We therefore removed replies (i.e., messages that answer a previous question from technical support), signatures and e-mail headings (except the `from` and `subject` fields). The analysis corpus contains an average of 47 words per message. This average is smaller than the Reuter-21578 corpus often used in text categorisation [2] (129 words), but larger than the typical questions of the QA track of TREC-9 (6.9 words) and TREC-8 (9.8 words) [25,26].

Determining the factual content of an e-mail is not sufficient to answer it correctly – its purpose is also very important. Typically, the identification of a text's purpose is not performed in IE, because the texts used in this task all share a single pre-determined purpose. In a newswire report about a terrorist attack, for example, the purpose of the text is to report factual data regarding events that have occurred. The same data can be extracted from a threat letter. In the latter case,

[1] Provided a textual knowledge base exists.
[2] www.research.att.com/~lewis.

the purpose of the text is much different, and factual data alone are insufficient to capture the text's meaning. In the e-mails we are analysing, the same set of data can be extracted from e-mails that have different purposes. The answer to be generated for these emails must therefore be different. In the analysis corpus, 4 types of purposes have been identified, each requiring a different type of answer.

Problem reports (67%): For example, reports that a printer is out of paper, a user cannot print a particular file, the printer is unreachable, etc. Here, the sender wants the problem fixed and wishes to receive an acknowledgement.

How-to questions (19%): For example, questions about how to print on both sides of the paper, how to kill a job, etc. Here, the expected response is a set of specific instructions to solve the specific problem.

Request for general information (13%): For example, questions regarding properties of the printers (name, location, resolution, etc.). The sender wishes to receive information that does not relate to a specific problem.

Suggestions (1%): Suggestions are not frequent but do occur in our analysis corpus. Here, the sender does not necessarily expect an answer, so both a generic acknowledgement or a customised answer can be sent.

Fig. 1 shows examples of simple e-mails from our corpus and their English translation (for illustration purposes). Note that for confidentiality reasons, names of persons have been changed.

3. General architecture

The typical task of answering e-mail can be decomposed into three steps [4]: recognising the problem(s) (reading and understanding the e-mail); searching for a solution (identifying predefined text blocks) and providing a solution (customising the text blocks and sending the text). Our interest lies in the first and last steps: understanding the text and formulating the response.

The general architecture of the system is shown in Fig. 2. As a printer-related message arrives, IE tries to fill pre-defined extraction templates that are then passed to a knowledge-intensive, domain-dependent process that checks the validity of the extracted information. Valid templates are further filled by inferring new information from the extraction templates and a domain knowledge base. Depending on the template content, a set of answer specifications is selected, filled and organised by an NLG module. The emphasis of the work is on the IE and the NLG modules (modules in grey in Fig. 2).

When responding to an e-mail, four situations can occur:

Answer found: The extraction templates contain correct information and the decision process can find a suitable answer specification. In this case, the extraction and the answer specification are passed to the generation module.

Human referral: The extraction templates contain correct information but the decision process cannot find a suitable answer. In this case, a *Human referral* message is produced. [3]

[3] Our interest lies in answering the e-mail and not in actually routing it to a clerk or department. A technique based on text classification or case-based reasoning may be used to select the most appropriate clerk to whom the e-mail should be routed.

Problem report	
From: Jane Black <black@iro.umontreal.ca> Subject: toner imprimante Bonjour, Est-ce que le toner de l'imprimante hp2248 peut etre remplace par un neuf? L'impression est de mauvaise qualite et j'ai besoin d'imprimer des documents pour une conference vendredi.	*From: Jane Black <black@iro.umontreal.ca>* *Subject: toner printer* *Hi,* *Can the toner [cartridge] on printer hp2248 be replaced by a new one? The printing is of bad quality and I need to print documents for a conference Friday.*

How-to question	
From: David Smith <smith@iro.umontreal.ca> Subject: Bonjour, J'aimerais savoir comment je peux imprimer seulement sur un côte de la page sur l'imprimante hp2248. Merci. David	*From: David Smith <smith@iro.umontreal.ca>* *Subject:* *Hi,* *I would like to know how I can print on one side of the page only on the hp2248 printer.* *Thank you.* *David*

Request for general information	
From: John Doe <doe@iro.umontreal.ca> Subject: j'ai envoye un message sur l'imprimante (a partir de elm sur champlain) apres lpq j'ai vu que l'imprimante s'appelle hp4sialdus. Pouvez vous me dire ou se trouve cette imprimante? JD	*From: John Doe <doe@iro.umontreal.ca>* *Subject:* *i sent a message to the printer (from elm on champlain) after lpq I saw that the printer is called hp4sialdus. Can you tell me where this printer is located?* *JD*

Suggestions	
From: Jane Black <black@iro.umontreal.ca> Subject: Ajout a IRO.FAQ Cher support, L'option -i-1, pour imprimer recto seulement sur les imprimantes HP, tel que decrit dans <<IRO.FAQ>>, est sans e- et depuis capchat, en fait, je n'ai obtenu le recto que depuis saguenay. Il faudrait, a mon avis, en faire mention dans le FAQ ou encore faire en sorte que l'option fonctionne de toutes les stations.	*From: Jane Black <black@iro.umontreal.ca>* *Subject: Addition to IRO.FAQ* *Dear technical support,* *The option -i-1, to print on one side only on the HP printers, as described in <<IRO.FAQ>>, does not work from capchat, in fact, I was only able to print single-sided from saguenay. I think that either the FAQ should mention this or the option should be available from all machines.*

Fig. 1. Examples of questions from our corpus and their English translation.

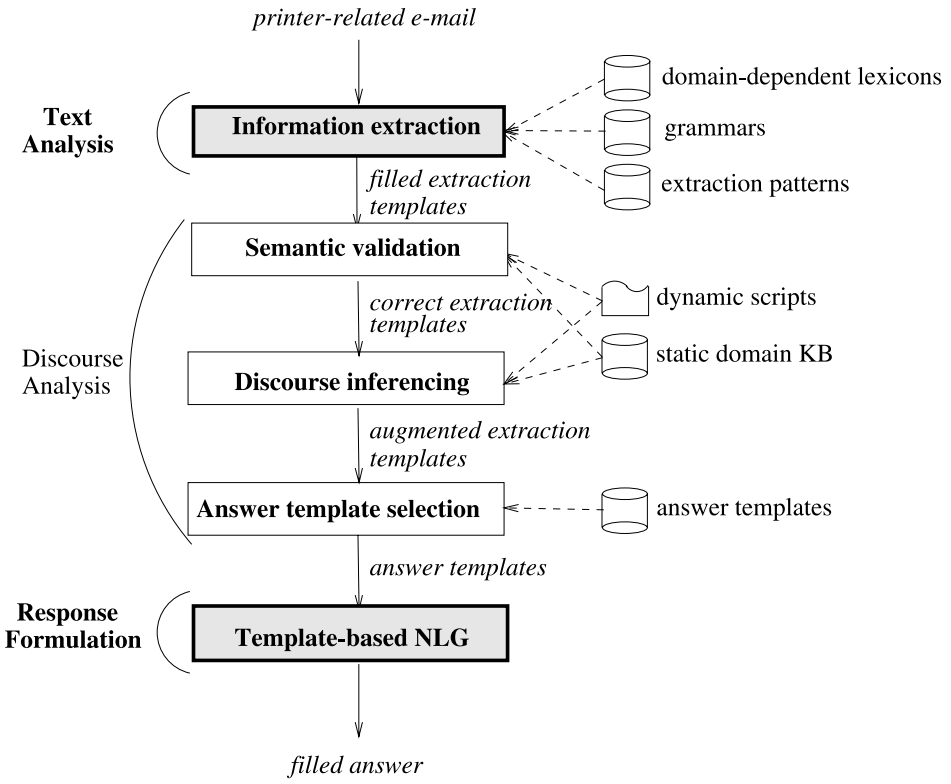

Fig. 2. Architecture of the system.

Incorrect information: The extraction templates contain incorrect or incoherent information. This situation can arise from a legitimate error made by the sender in the message, or from an error in the extraction module. Because the source cannot be determined, in both cases a generic *Incorrect information* message is generated.

Incomplete information: The extraction templates do not contain enough material to select an answer specification. This can occur if the message does not contain the necessary information, or if the extraction module does not find it. In this case, a message of the type *Missing information* is generated. Note that no conversation management is performed to wait for and recover the missing information in subsequent messages.

Let us now describe each process in detail.

3.1. Text analysis

The task of an IE system is to identify specific information from a natural language text in a specific discourse domain and to represent it in a structured template format [21–23]. For example, from a report on management succession, an IE system will be able to identify who replaced whom, on what date, in what post, in what organisation, and so on. This was precisely the goal of the MUC-6 conference [22]. The filled templates can then be stored in a database for later retrieval or serve as a basis for the automatic generation of summaries.

IE from formal texts usually follows one of two approaches: a linguistic surface approach or a statistical approach. The linguistic approach is based on a lexico-syntactic description of the phrases to be located [1,2,14]. With this approach, the text is tokenised, each token is tagged with its most likely grammatical category, and syntactic chunking is performed to group together noun and verb phrases. Next, lexico-syntactic extraction patterns and large dictionaries of trigger phrases (*M., inc.*, etc), of known proper names and of general language are used to identify and semantically tag key phrases. Discourse rules are then applied to relate this key information and to infer new facts.

To account for noise in certain kinds of texts, IE is often performed by statistical methods which use a language model trained on large pre-tagged corpora [13,20]. Studies have shown that the probabilistic methods yield good results if large corpora are available for training. However, to apply the system to different discourse domains, retraining of the model is necessary. In contrast, linguistic rule-based systems can be tuned more easily from a small corpus. Because our corpus was so small and we already had a rule-based IE prototype for French [16], we followed a rule-based approach. As is typically done, we tokenise and lemmatise the texts and make use of lexicons, grammars and extraction patterns.

For each printer-related message, the IE module tries to fill four types of templates:

- *Message template*: describing the message itself (rather than its content); its purpose, sender, etc.
- *Template element*: describing named entities [24]. In our application, templates describing users, files, printers, and so forth are filled.
- *Template relation*: capturing general relations between template elements [6]. Relations such as `userl` *is the owner of* `filel` are represented.
- *Scenario template*: capturing domain- and task-specific relations [24]. In our application, the scenario defines a print event, that is, a user trying to print a file on a printer.

The filled templates are used to represent the content of the printer question and to find an appropriate answer. While the design and extraction of some fields are domain independent (e.g., person names) and can use publicly available resources, others are domain dependent (e.g., printer or file templates) for which specific grammars and lexicons must be built. Fig. 3 shows some examples of extraction templates. Each field value can either be a free-form string extracted from the document (e.g., `David Smith`), a value from a closed set of possible answers (e.g. a printer name) or a pointer to another template. In the current state of the system, the message template is not filled automatically, but an n-gram key-word approach as used by [15] is envisaged. Template elements, scenarios and template relations are filled using three techniques:

Lexicons: This includes lists of known users, laboratories, software and printer names built from the technical support databases.

Grammars of named entities: This includes such things as regular expressions for recognising file names from unknown words.

Lexico-syntactic extraction patterns: This includes patterns for recognising named entities from their neighbouring words. For example, the pattern `printer::name` [4] `in room X` allows us to infer that X is the room number of `printer::name` although it may not follow the grammar of

[4] The notation `X::Y` refers to field `Y` of template `X`.

message template	
Field	*Value*
purpose	set
topic	set
sender	user template
...	...

user template	
Field	*Value*
name	string
e-mail address	set
laboratory	set
...	...

print-event template	
Field	*Value*
sender	user template
destination	printer template
file_to_print	file template
source	machine template
...	...

printer template	
Field	*Value*
name	set
room	set
status	set
file_printing	file template
...	...

file template	
Field	*Value*
name	string
current_format	set
desired_format	set
current_page_size	set
desired_page_size	set
owner	user template
job_number	string
...	...

Fig. 3. Examples of extraction templates to be filled.

room numbers. Such patterns allow us to be somewhat flexible and recognise named entities that are not well formed.

Since our e-mails are factual and short and because the discourse domain is very specific, template relation extraction can be kept simple. As Fig. 3 shows, in most cases, the template elements can only have one relation with other templates. For example, any printer template is assumed to be the job destination (where the user wants to print) and machine templates are always assumed to be the job source (the machine from which the printing job is sent). In the cases of file and user templates, two relations are possible. In these cases, lexico-syntactic patterns are used to disambiguate between relations. For example, if a template entity for files is filled then it can be a `file_to_print` or a `file_printing`. To identify the correct relation, patterns such as `I wish to print file::name` or `file::name is blocking the queue` are used. Template relations are identifiable this way because the discourse domain is very specific, and the texts are factual and short.

Template and field coreference allow us to merge several templates if they refer to the same real-world entity. Several manually coded and automatically learned techniques exist to perform coreference resolution (e.g., [5]). Currently, coreference resolution has not specifically been addressed. We use the strategies of our existing IE system, EXIBUM [16], which merges entities only on the basis of head noun equality. This strategy allows us to determine that the names `David Smith` and `David` refer to the same person and that `smith@iro.umontreal.ca` is the e-mail

message 1	
purpose	how-to question
sender	user1 template
...	...

user 1	
name	David Smith
e-mail address	smith@iro.umontreal.ca
laboratory	∅
...	...

print-event 1	
sender	user1 template
destination	printer1 template
file_to_print	file1 template
source	∅
...	...

printer 1	
name	hp2248
room	∅
status	∅
file_printing	∅
...	...

file 1	
name	∅
current_format	∅
desired_format	single_sided
current_page_size	∅
desired_page_size	∅
owner	∅
job_number	∅
...	...

Fig. 4. Examples of extraction templates after information extraction.

address of this same person. However, this technique cannot determine that `JD` in the signature of the third message of Fig. 1 refers to `John Doe`. Fig. 4 shows the templates filled by discourse inferencing.

3.2. Discourse analysis

3.2.1. Semantic validation

Once the templates are filled, the field values should be validated. This process serves two purposes: making sure the user communicated correct information, so that a correct answer can be formulated, and more importantly, making sure the IE performed a correct analysis of the text. Extraction templates contain two types of field values: those used to select an answer specification, and those that do not influence the choice of answer specifications but rather are used to customise the answer. The latter fields are not verified, while the former are checked for correctness. Semantic validation is performed through two strategies:

1. Matching filled fields against one another and against a static knowledge base to verify their coherence. For example, if the IE module is extracted within the same printer template the fields `printer1::name=hp2248` and `printer1::room=X-234`, but the database does not contain the pair `(name=hp2248, room=X-234)`, then an incoherence is detected.
2. Running dynamic scripts associated with specific template fields. For example, if the IE module is extracted within the same printer template the fields `printer1::name=hp2248` and `printer1::file_printing::name=test.txt`, but the script associated with printer names to find the name of the currently printing file (namely the command `lpq -P printer::name`) has determined that printer hp2248 is currently printing another file, then an incoherence is detected.

user 1	
name	David Smith
e-mail address	smith@iro.umontreal.ca
laboratory	**artificial intelligence**
...	...

printer 1	
name	hp2248
room	**P-204**
status	**printing**
file_printing	**file 2**
...	...

file 1	
name	∅
current_format	∅
desired_format	single_sided
current_page_size	**8.5x11**
desired_page_size	**8.5x11**
owner	∅
job_number	∅
...	...

file 2	
name	**test.txt**
current_format	∅
desired_format	∅
current_page_size	**8.5x11**
desired_page_size	**8.5x11**
owner	∅
job_number	∅
...	...

Fig. 5. Examples of extraction templates after discourse analysis.

Incorrect templates are flagged and are sent directly to the answer selection, that will select an *Incorrect information* type of answer. On the other hand, correct templates are further filled by discourse inferencing (see Fig. 5).

3.2.2. Discourse inferencing

The role of discourse inferencing is to infer information or relations not explicitly stated in the text, but known from the discourse itself or the domain. Discourse inferencing tries to further fill extraction templates and create new ones. This analysis is performed also by the use of static and dynamic knowledge bases, and by the use of default values for empty fields. For example, the fact that `David Smith` is a member of the `artificial intelligence` laboratory can be determined by matching the e-mail address `smith@iro.umontreal.ca` to the static knowledge base. Also, a dynamic script can determine that printer `hp2248` is currently printing a file named `test.txt`. Because this file is never mentioned in the e-mail message, a new `file` template can be created (see Fig. 5).

3.2.3. Answer template selection

Selecting the generic content of the response is done using a decision tree developed specifically for this discourse domain. Conditions of the decision tree relate to field values of the extraction templates, while the leaves of the decision tree are answer specifications (see Fig. 6). Answer specifications can contain canned text to be used as is, but can also contain command names, options, syntax, special considerations and a location where to find more information, for which the formulation can vary.

For example, a value for the field `file_to_print::desired_format` indicates that the topic of the e-mail is to print a file in a particular format and the corresponding decision tree is traversed. If the extraction templates do not contain enough information to reach an answer specification (a leaf), then an *Incomplete information* answer is selected. If no answer specification can be reached because of invalid information in the extraction templates, a *Human referral* (no

Answer specification 1	
canned text:	*To print on only one side of the paper [on printer* `printer::name`*], use the command* `answer::command` *[with the* `answer::option` *option]. Type:* `answer::syntax`.
command:	*lpr*
option:	*-i-1*
syntax:	*lpr -P* `printer::name answer::option file::name`
special considerations:	*Note that this printing mode should be used only for the final copy of a document.*
more info:	www.theURL/lpr#recto

Answer specification 2	
canned text:	It seems to be a printer hardware problem. A technician will address it shortly.

Fig. 6. Examples of answer specifications.

answer found) is selected. Finally, if an answer specification is reached, it will be customised by the template-based NLG module.

3.3. Response formulation

Once an answer specification is selected, it must be filled and organised into a cohesive answer. To produce a response, the pre-defined rhetorical schemas of Fig. 7 are followed. Regardless of which situation we are dealing with, the response will contain welcome greetings, a message body (Incorrect info or Missing info or Human referral or a Customised response), closing greetings and a signature. A repertoire of canned answers is available to produce each part of the response. Some canned answers are fully lexicalised, while others contain slots to be filled by information contained in the extraction templates or the answer specification. For example, welcome greetings can simply be `Hello` or `Dear sender::name`. If the extraction templates contain the name of the sender, `Dear sender::name` will be preferred over `Hello`. If several equivalent variants of the same part of the answer have been specified, such as `Dear sender::name` and `Hi sender::name`, one of the variants is randomly selected by the system.

Welcome greetings
Incorrect info | Missing info | Human referral | Customised response
Closing greetings
Signature

(a) All responses

Answer
[Special Considerations]
[Location for more info]

(b) Customised responses

Fig. 7. Rhetorical schemas for responses.

Generated Answer	English Translation
Bonjour David, Pour imprimer en recto seulement sur la hp2248, il faut utiliser la commande lpr avec l'option -i-1. Faire: lpr -P hp2248 -i-1 <nom du fichier> Notez que ce mode d'impression ne doit être utilisé que pour la copie finale d'un document. Pour plus d'information, consutez l'URL: www.theURL/lpr#recto Bonne chance, Support technique	*Hi David,* *To print on only one side of the paper on the hp2248, you must use the command lpr with the option -i-1. Type: lpr -P hp2248 -i-1 <file name>* *Note that this printing mode should only be used for the final copy of a document.* *For more information, consult the URL: www.theURL/lpr#recto* *Good Luck,* *Technical support*

Fig. 8. Generated answer for the how-to question of Fig. 1.

The process of deriving the surface form from the answer specification is carried out in two steps. A Prolog definite clause grammar (DCG), specifying the rhetorical schema, the canned text and parameterised slots, and various helper predicates and rules, are used to generate the text level of abstraction [12], that is the full sequence of words and phrases in the response. The final orthographic form is then generated from this sequence of words. Issues such as elision, contraction, punctuation, capitalisation and formatting are handled in this step by the motor realisation module of the SPIN generation system [17].

The customised response is built from the filled answer specification selected by the decision tree. ''Special considerations'' and ''Location for more information'' are optional parts of the response schemas and will be filled depending on the level of detail desired in the responses. The output for the how-to question of Fig. 1 is shown in Fig. 8 along with an English translation.

4. Implementation and results

The system we have described is currently under development, but a working prototype has been built to assess the feasibility of the project. In its current state, the IE module is not complete, the validation and inferencing modules have not been implemented as the IE is not complete, but the answer selection and the NLG module are in working state.

An evaluation of the answer selection module has been performed. The goal of the evaluation was to determine the coverage of the decision module and the associated response templates. The evaluation was performed by two assessors on answers generated for the questions of the test corpus. To isolate the evaluation of the decision module, extraction templates have been filled by hand [5] and

[5] The criteria used to fill in the extraction templates by hand was to provide values that could reasonably be extracted automatically using the lexico-syntactic approach described above. This allowed for errors and incomplete extraction templates.

Table 1
Evaluation of the content of the generated answers

	Correct	Incorrect	Total
Answer found	27.7%	13.3%	41.0%
Human referral	38.7%	20.3%	59.0%
Incorrect information	–	–	–
Incomplete information	–	–	–
Total	66.4%	33.6%	100%

given as input to the decision module to generate an answer. Two human assessors read the test questions and evaluated the correctness of the content of the generated answers (not its linguistic quality) as compared with the original responses given by the technical support staff. An answer was considered correct if it was similar in content to the technical support's answer or if it was an alternative precise and responsive answer. Therefore, answers that were counted as incorrect include those that contain a specific but wrong answer and those that contain a human referral for which technical support gave a specific answer. The results are shown in Table 1. On average 66.4% of the evaluation questions have been answered correctly. Although 33.6% incorrect answers seems high, among the incorrect answers 13.3% are answers whose content is actually wrong, while 20.3% are unnecessary human referral answers. Note that the *incorrect* or *incomplete information* categories were not evaluated as there are no cases of these types in our corpus.

This evaluation showed that the template selection process still needs to be refined to account for the 13.3% wrong answers. More importantly though, the evaluation demonstrated that the general architecture of the system is viable. That is, the extraction templates and the answer selection decision module are sufficient to generate automatic responses to e-mails on a specific discourse domain.

Our priority now is to finish the implementation of the prototype so that a formal evaluation can be performed. Once the prototype will be in place, we plan to evaluate it using 3 measures: a measure of the IE module, a measure of the linguistic quality of the NLG module and a global measure combining the two. Measuring the IE will be done using the MUC evaluation protocol [7] on the test corpus. Measuring the NLG module will be done through a blind evaluation protocol similar to [9].

5. Related work

Related work on e-mail answering includes commercial e-mail answering systems and research in QA and e-mail classification.

The simplest level of e-mail answering systems is the so-called *auto-responder*. [6] These systems return a canned document in response to an e-mail according to the presence of keywords in the subject or body of the message. A variant of auto-responders can customise the returned document if the user fills in a predefined Web form. An obvious drawback of these systems is that they

[6] Also known as *AR*, *infobots*, *mailbots* or *e-mail-on-demand*.

do not analyse the content of free-form messages. More sophisticated types of e-mail responder are included in e-mail management systems, and can provide pre-written response templates for frequently asked questions. Slots are usually filled in with information extracted manually from the incoming mail, but some systems seem to perform the extraction automatically [3,19].

QA tries to find an answer to a natural language question [25,26] from a large set of documents. The question type is determined by the presence of trigger phrases (e.g. *where*, *how many*, *how much*), which indicates the type of the answer required (e.g. *location*, *number*, *money*). IR is typically performed to identify a subset of the documents and a set of passages that may contain the answer. Named entities are then extracted from these passages and semantically tagged and the string containing the best scoring entity is retained as the answer. QA differs from e-mail answering in several aspects. Generally speaking, e-mail answering is interested in *analysing* a longer text and *formulating* a linguistically motivated answer, while QA takes a short and explicit question as input and focuses on *locating* the answer. Issues in discourse analysis must therefore be addressed in e-mail answering, but not in QA. In addition, questions in QA are, for the moment, restricted to specific types such as *who*, *why*, *where* but pertain to an unrestricted discourse domain. On the other hand, in e-mail answering, the questions are of unrestricted type, but the discourse domain is typically restricted.

E-mail classification is another domain related to our work that has been addressed by many research projects. This has been approached both from an automatic learning perspective (e.g. [4,11]) and from an IE perspective (e.g. [8]). Our work complements those in text classification as it supposes that the incoming e-mail messages have already been classified as printer-related questions. E-mail classification can therefore be seen as a pre-processing module to our system.

On the NLG side, Coch developed a system to generate automatic answers to complaint letters from clients of La Redoute (a large French mail-order corporation) [9,10]. As letters are not in electronic format, the reading, the extraction and the decision is done by humans, but the production of a well-formed response is done automatically. Through a formal blind evaluation, Coch has demonstrated that the best responses (according to specific criteria) are still the human-generated ones, but that the use of a hybrid template-based NLG system produced acceptable responses at a much faster rate.

6. Discussion and further research

In this paper, we have described the design of an e-mail answering system we are currently developing. The system relies on information extraction to *analyse* the user message, a decision tree to determine the content of the answer, and template-based natural language generation to produce the surface form of the answer in a customised and cohesive way. Because the discourse domain is specific and high precision scores are desired, a knowledge-intensive, domain-dependent approach is used. Although this type of system offers poor adaptability, it was viewed as a means to reach high precision scores in text analysis. This is crucial in e-mail answering, as a wrong answer sent to a client can have far-reaching customer satisfaction consequences. The approach is therefore much more costly than IR from an FAQ, which requires no specifically built knowledge base. If customisation of the answers and high precision are not required, IR from a FAQ is an interesting alternative. In order to scale up the approach to larger domains, we believe that new domain-dependent knowledge bases, extraction rules and answer specifications should be

developed and that text classification should be performed prior to IE in order to select the appropriate knowledge bases to use. However, we expect that enlarging the discourse domain will decrease precision because of lower confidence levels in the extracted templates (due to the IE itself or discource inferencing) coupled with a more complex answer selection decision tree.

So far, we have not taken into account how textual noise from the e-mail affects the textual analysis. E-mail messages are informal electronic texts that do not follow strict writing guidelines. Textual noise can come from typography (e.g., punctuation, lack of diacritics and capitalisation), terminology (e.g., informal abbreviations), and orthographic and grammatical irregularities. Our approach is based on the MUC experiences that have mainly been concerned with homogeneous corpora that follow writing guidelines. The rule-based keyword approach may need to be adapted to account for textual noise.

For the moment, the NLG system fills answer slots directly, without much linguistic knowledge. We plan to increase the cohesiveness and naturality of the responses by using referring expressions whenever possible. The work of Kosseim et al. [18], for example, provides linguistically motivated guidelines for the generation of such expressions in French.

Another avenue of interest is the writer's communicative goal. We have assumed that the communicative goal of e-mail messages is to be informative and factual. The text analysis has thus not taken into account negation, polarity, and rhetoric. Tone, irony or humour can be used in messages, and if wrongly interpreted, can lead to adverse effects. It would be interesting to be able to detect the communicative goal of the e-mail and to be able to produce a *Human referral* response to non-factual e-mails.

Acknowledgements

We are grateful to our colleagues from RALI and to the anonymous referees for their relevant comments. This work was supported by a grant from Bell University Laboratories.

References

[1] J. Aberdeen, J. Burger, D. Day, L. Hirschman, P. Robinson, M. Vilain, MITRE: Description of the Alembic system as used for MUC-6, in: MUC-6 [22].
[2] D. Appelt, J. Hobbs, J. Bear, D. Israel, M. Kameyama, M. Tyson, SRI: Description of the JV-FASTUS system used for MUC-5, in: MUC-5 [21].
[3] www.brightware.com, Site visited in December 2000.
[4] S. Busemann, S. Schmeier, R. Arens, Message classification in the call center, in: Proceedings of ANLP-2000, Seattle, 2000, pp. 159–165.
[5] C. Cardie, K. Wagstaff, Noun phrase coreference as clustering, in: Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 1999, pp. 82–89.
[6] N. Chinchor, E. Marsh, MUC-7 information extraction task definition (version 5.1), in: MUC-7 [23], available at http://www.itl.nist.gov/iad/894.02/related_projects/muc/proceedings/muc_7_toc.html
[7] N. Chincor, G. Dungca, Four scorers and seven years ago: the scoring method for MUC-6, in: [22], pp. 33–38.
[8] F. Ciravegna, A. Lavelli, N. Mana, J. Matiasek, L. Gilardoni, S. Mazza, M. Ferraro, W. Black, F. Rinaldi, D. Mowatt, Facile: classifying texts integrating pattern matching and information extraction, in: Proceedings of IJCAI-99, Stockholm, Sweden, 1999, pp. 890–895.
[9] J. Coch, Evaluating and comparing three text-production techniques, in: Proceedings of COLING-96, Copenhagen, Denmark, 1996.

[10] J. Coch, J. Magnoler, Quality tests for a mail generation system, in: Proceedings of Linguistic Engineering, Montpellier, France, 1995.
[11] W. Cohen, Learning rules that classify e-mail, in: Proceedings of the 1996 AAAI Spring Symposium on Machine Learning in Information Access, 1996.
[12] R. Dale, E. Reiter, in: Building Natural Language Generation Systems Studies in Natural Language Processing, Cambridge University Press, New York, 2000.
[13] L. Dekan, Using collocation statistics in information extraction, in: MUC-7 [23], available at http://www.itl.nist.gov/iad/894.02/related_projects/muc/proceedings/muc_7_toc.html
[14] R. Grishman, Where's the syntax? The NYU MUC-6 system, in: MUC-6 [22].
[15] H. Khosravi, Y. Wilks, Routing email automatically by purpose not topic, Natural Language Engineering 9 (3) (1999) 237–250.
[16] L. Kosseim, G. Lapalme, Exibum: un système expérimental d'extraction d'information bilingue, in: Rencontre Internationale sur l'extraction le filtrage et le résumé automatique (RIFRA-98), Sfax, Tunisia, November 1998, pp. 129–140.
[17] L. Kosseim, G. Lapalme, Choosing rhetorical structures to plan instructional texts, Computational Intelligence: An International Journal 16 (3) (2000) 408–445.
[18] L. Kosseim, A. Tutin, R. Kittredge, G. Lapalme, Generating grammatical and lexical anaphora in assembly instruction texts, in: G. Adorni, M. Zock (Eds.), Trends in Natural Language Generation, Springer, Berlin, 1996, pp. 260–275.
[19] Y. Lallement, M. Fox, Interact: a staged approach to customer service automation, in: H. Hamilton, Q. Yang (Eds.), Canadian AI 2000, Springer, Berlin, 2000, pp. 164–175.
[20] D. Miller, R. Schwartz, R. Weischedel, R. Stone, Named entity extraction from broadcast news, in: Proceedings of DARPA Broadcast News Workshop, Herndon, Virginia, 1999.
[21] Proceedings of the Fifth Message Understanding Conference (MUC-5), Morgan Kaufmannn, Baltimore, MD, November 1995.
[22] Proceedings of the Sixth Message Understanding Conference (MUC-6), Morgan Kaufmann, Columbia, MD, November 1995.
[23] Proceedings of the Seventh Message Understanding Conference (MUC-7), 1999, available at http://www.itl.nist.gov/iad/894.02/related_projects/muc/proceedings/muc_7_toc.html
[24] B. Sundheim, Overview of Results of the MUC-6 Evaluation, pp. 13–31.
[25] Proceedings of the Text Eighth REtrieval Conference (TREC-8), Gaithersburg, MD, November 1999.
[26] Proceedings of the Ninth Text REtrieval Conference (TREC-9), Gaithersburg, MD, November 2000.

Leila Kosseim is a researcher at the RALI, the laboratory for applied research in computational linguistics, at the Université de Montréal since 1998. From 1995 to 1998, she worked in R&D at Druide informatique inc, developing linguistic software tools. She obtained her M.Sc. and Ph.D. in Natural Language Generation from the Université de Montréal in 1992 and 1995.

Stéphane Beauregard has a B.Sc. (Physics, Carleton University) and ten years industrial experience as a programmer/analyst. He will soon complete his M.Sc. in Computer Science at the Université de Montréal on the topic of text generation. His research interests are Machine Learning and Natural Language Processing.

Guy Lapalme is a professor in Computer Science at the Université de Montréal, and a member of the RALI laboratory. He has been working in the field of Natural Language Processing for more than 15 years, especially in Natural Language Generation. More recently he became involved in Machine Aided Translation and Information Extraction.