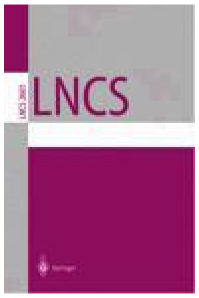
Book Chapter



Using Information Extraction and Natural Language Generation to Answer E-Mail

Book Series	Lecture Notes in Computer Science
Publisher	Springer Berlin / Heidelberg
ISSN	0302-9743
Subject	Computer Science
Volume	Volume 1959/2001
Book	Natural Language Processing and Information Systems: 5th International Conference on Applications of Natural Language to Information Systems, NLDB 2000, Versailles, France, June 2000. Revised Papers
Copyright	2001
Page	152

Authors

Leila Kosseim, Stéphane Beauregard, Guy Lapalme

Abstract

This paper discusses the use of information extraction and natural language generation in the design of an automated e-mail answering system. We analyse short free-form texts and generating 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.

Open: Entire document

Using Information Extraction and Natural Language Generation to Answer E-Mail

Leila Kosseim, Stéphane Beauregard and Guy Lapalme

RALI, DIRO, Université de Montréal
CP 6128, Succ. Centre Ville, Montréal (Québec) Canada, H3C 3J7
{kosseim, beaurs, lapalme}@iro.umontreal.ca

Abstract

This paper discusses the use of information extraction and natural language generation in the design of an automated e-mail answering system. We analyse short free-form texts and generating 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.

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 NLP techniques very attractive. E-mail messages make up a large portion of the free-form documents that are currently treated manually. 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 and natural language generation 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 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 to develop the technology.

Unlike typical question answering systems (e.g. [20]) 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: information retrieval or information extraction. With the

[0] source: Proceedings of the 5th International Conference on Applications of Natural Language to Information Systems (NLDB'2000). june 2000. Versailles, France

information retrieval approach, the incoming message is considered as a query to be matched against some textual knowledge base (e.g. a 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. It 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), information extraction seems in our view more adequate for analysing the incoming e-mail and template-based natural language generation 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 done in French. We have concentrated our efforts on a specific discourse domain in order to obtain good results in information extraction and to be able to manage knowledge representation. The entire corpus covered a 3 year period and is composed of 188 e-mails. The corpus was split into two sets: the first 2 years for analysis and the last year for testing. 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 final analysis corpus contains 126 messages with an average of 47 words per message. This average is smaller than the Reuter-21578 text categorisation test collection [2] (129 words), but larger than the typical questions of the QA track of TREC-8 (9 words[3]) [20]. The messages from the analysis corpus fall into 3 major query types:

- problem reports (67%): For example, reports that a printer is out of paper, a user can't print a particular file, the printer is unreachable, ...
- how-to questions (19%): For example, questions about how to print on both sides of the paper, how to kill a job, ...
- general information (13%): For example, questions regarding properties of the printers (name, location, resolution, ...)

[1] Provided a textual knowledge base exists
[2] www.research.att.com/~lewis
[3] average of the NIST 38 development questions (www.research.att.com/~singhal/qa-dev-set)

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

Fig. 1. Example of a how-to question from our corpus and its English translation

Figure 1 shows a example of a simple e-mail and its 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 3 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 Figure 2. As a printer-related message arrives, information extraction 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 templates is selected, filled and organised by a natural language generation module. The emphasis of the work is on the information extraction and the natural language generation modules (modules in bold in Figure 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 template. In this case, the extraction and the answer templates 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.[4]

Incorrect information: The extraction templates contain incorrect or incoherent information. This situation can arise from a legitimate error made

[4] 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 route the e-mail to.

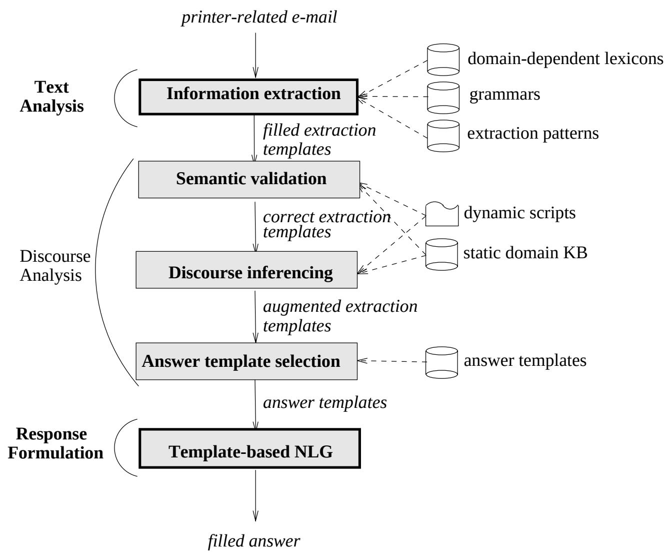


Fig. 2. Architecture of the system

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 template. This can occur if the message did not contain the necessary information, or if the extraction module did 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.

3.1 Text Analysis

The task of an Information Extraction (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. For example, from a car accident report, an IE system will be able to identify the date and location of the accident, and the names and status of the victims. 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, 12]. 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.*, ...), 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 ones. To account for noise in certain kinds of texts, IE is often performed by statistical methods which uses a language model trained on large pre-tagged corpora [11, 17]. 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 information extraction prototype for French [13], 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.

In our project, the IE module tries to fill a template relation and a set of template elements for each printer-related message. The filled templates will be used to diagnose the printer question and find an appropriate answer. Following the MUC terminology [18], template elements are templates describing named entities (e.g. templates for persons, artifacts, ...) and templates relations represent relations between template elements (e.g. `person1` *is the owner of* `artifact1`). While the design and extraction of some fields are domain independent (e.g. person templates, location templates) 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. In our discourse domain, the system tries to fill templates for computer users, printers, files, machines and actions performed. These templates are shown in Figure 3. 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. Template entities are filled through 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 neighboring words. For example, the pattern `printer::name`[5] `in room X` allows us to infer that X is the room number of `printer::name` although it may not follow the grammar of room numbers. Such patterns allows us to be somewhat flexible and recognise named entities that are not well formed.

Because our e-mails are factual and short and because the discourse domain is very specific, template relation extraction can be kept simple. As Figure 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

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

(where the user wants to print) and machine templates are always assumed to be the job source (the machine from where 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.

print event template		
Field	*Value*	*Validation*
sender	user template	no
destination	printer template	no
source	machine template	no
file_to_print	file template	no
action_tried	action template	no

user template		
Field	*Value*	*Validation*
name	string	no
e-mail address	set	yes
laboratory	set	yes

printer template		
Field	*Value*	*Validation*
name	set	yes
room	set	yes
status	set	yes
file_printing	file template	yes

file template		
Field	*Value*	*Validation*
name	string	yes
current_format	set	yes
desired_format	set	yes
current_page_size	set	yes
desired_page_size	set	yes
generated_how	set	yes
owner	user template	no
job number	string	no

machine template		
Field	*Value*	*Validation*
name	set	yes
OS	set	yes
laboratory	set	yes

action template		
Field	*Value*	*Validation*
comand_tried	string	yes
error_message	string	yes

Fig. 3. Extraction Template Relation and Template Entities to be filled

Template and field coreference allows 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]). In our system, coreference resolution has not specifically been addressed. We use the strategies of our existing IE system, EXIBUM [13], 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 address of this same person. However, this technique cannot determine that `JD` in the signature of the second message of Figure 1 refers to

`John Doe`. Figure 4 shows the templates filled by the IE module for the how-to question of Figure 1.

print event 1	
sender	user1 template
destination	printer1 template
source	
file_to_print	file1 template
action_tried	

printer1 template	
name	hp2248
room	
status	
file_printing	

file1 template	
name	
current_format	
desired_format	single_sided
current_page_size	
desired_page_size	
generated_how	
owner	
job number	

user1 template	
name	David Smith
e-mail address	smith@iro.umontreal.ca
laboratory	

Fig. 4. Extraction Templates after Information Extraction

3.2 Discourse Analysis

Semantic Validation and Discourse Inferencing Once the templates are filled, the field values are 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 information extraction performed a correct analysis of the text. Extraction templates contain two types of field values: those used to select an answer template, and those that do not influence the choice of answer templates 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 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 to specific template fields. For example, if the IE module 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, `lpq -P printer::name`) has determined that printer hp2248 is currently printing another file, then an incoherence is detected.

Incorrect templates are flagged and are sent directly to the answer template selection, that will select an *Incorrect information* type of answer. On the other hand, correct templates are further filled by discourse analysis.

The role of discourse analysis is to infer information or relations that is not explicitly stated in the text, but that are known from the discourse itself or the domain. Discourse analysis tries to further fill extraction templates and create new ones. This analysis is performed also by the use of a static and a 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; while a dynamic script can determined that printer `hp2248` is currently printing the file `test.txt`, thus creating a new `file` template.

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 templates (see Figure 5). Answer templates can contain canned answers to be used as is, but can also contain command names, options and 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 template (a leaf), then an *Incomplete information* answer template is selected. If no answer template can be reached because of unpredicted information in the extraction templates, a *Human referral* (no answer found) is selected. Finally, if an answer template is reached, it will be customised by the template-based Natural Language Generation (NLG) module.

canned answer:	∅
command:	lpr
option:	-i-1
syntax:	lpr -P destination::name -i-1 file_to_print::name
special considerations:	*Note that this printing mode should only be used for the final copy of a document.*
more info:	www.theURL/lpr#recto

Fig. 5. Example of an answer template

3.3 Response Formulation

Once an answer template is selected, it must be filled and organised into a cohesive answer. To produce a response, pre-defined rhetorical schemas 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 templates. 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.

The process of deriving the surface form from the answer template is carried out in two steps. A Prolog definite clause grammar (DCG), specifying the rhetorical schema, the canned text and parameterized slots, and various helper predicates and rules, is used to generate the text level of abstraction [10], 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 [14].

The customised response is built from the filled answer template 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 Figure 1 is shown in Figure 6 along with an 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'info, consutez l'URL: www.theURL/lpr#recto

[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 info, consult the URL: www.theURL/lpr#recto]

Bonne chance,

Support technique

Fig. 6. Generated answer for the how-to question of Figure 1

4 Related Work

Related work on e-mail answering include commercial e-mail answering systems, question answering and e-mail classification.

The simplest level of e-mail answering systems is the so-called *autoresponder*[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 autoresponders can customise the returned document, if the user filled in a predefined Web form. An obvious drawback of these systems is that they do not analyse the content of a free-form message. A more sophisticated type 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, 16].

Question answering (QA) tries to find an answer to a natural language question [20] form 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 (*location, number, money*, ...). Information retrieval 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: *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, 9]) and from an IE perspective (e.g. [6]). 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 developped a system to generate automatic answers to complaint letters from clients of LaRedoute (a large French mail-order corporation) [7, 8]. As letters are not in electronic format, the reading and extraction is done by humans, but the decision and the production of the 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] also known as *AR*, *infobots*, *mailbots* or *e-mail-on-demand*

5 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 approach is used. In order to scale up the approach to larger domains, we believe that new domain-dependent knowledge-bases, extraction rules and answer templates should be developped and that text-classification should be performed prior to IE in order to select the appropriate knowledge bases to use. Although the design of a knowledge-intensive domain-dependent system offers poor adaptability to other discourse domains, it was viewed as a means to reach high precision scores in text analysis; something that is crucial in e-mail answering, as a wrong answer sent to a client can have far-reaching customer satisfaction consequences. We believe the incremental approach to be appropriate; i.e. testing the precision of the system on a small discourse domain, and incrementally enlarging the discourse domain.

As the system is under development, no formal evaluation has yet been performed. As far as further research is concerned, our priority is therefore finishing the implementation of the prototype so that a formal evaluation can be performed. We plan to evaluate the system using 3 measures: a measure of the IE module, a measure of the NLG module and a global measure combining the two. Measuring the IE will be done using the MUC evaluation protocol [19] on the test corpus. Measuring the NLG module will be done through to a blind evaluation protocol similar to [7].

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. lack of diacritics and capitalisation), terminology (e.g. informal abbreviations), orthography 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 key-word 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. [15], for example, provides linguistically-motivated guidelines for the generation of such expressions in French.

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 the Laboratoire Universitaire Bell (LUB).

References

1. J. Aberdeen, J. Burger, D. Day, L. Hirschman, P. Robinson, and M. Vilain. MITRE: Description of the Alembic System as Used for MUC-6. In MUC-6 [19].
2. D. Appelt, J. Hobbs, J. Bear, D. Israel, M. Kameyama, and M. Tyson. SRI: Description of the JV-FASTUS System Used for MUC-5. In MUC-5 [18].
3. www.brightware.com/agents/answer.html. site visited in december, 1999.
4. S. Busemann, S. Schmeier, and R. Arens. Message Classification in the Call Center. In *Proceedings of ANLP-2000*, pages 159–165, Seattle, 2000.
5. C. Cardie and K. Wagstaff. Noun Phrase Coreference as Clustering. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 82–89, 1999.
6. F. Ciravegna, A. Lavelli, N. Mana, J. Matiasek, L. Gilardoni, S. Mazza, M. Ferraro, W. Black, F. Rinaldi, and D. Mowatt. Facile: classifying texts integrating pattern matching and information extraction. In *Proceeding of IJCAI-99*, pages 890–895, Stockholm, Sweden, 1999.
7. J. Coch. Evaluating and comparing three text-production techniques. In *Proceedings of COLING-96*, Copenhagen, Dannemark, 1996.
8. J. Coch and J. Magnoler. Quality tests for a mail generation system. In *Proceedings of Linguistic Engineering*, Montpellier, France, 1995.
9. W. Cohen. Learning rules that classify e-mail. In *Proceeding of the 1996 AAAI Spring Symposium on Machine Learning in Information Access*, 1996.
10. R. Dale and E. Reiter. *Building Natural Language Generation Systems.* Studies in Natural Language Processing. Cambridge University Press, 2000.
11. L. Dekan. Using Collocation Statistics in Information Extraction. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1999.
12. R. Grishman. Where's the Syntax? The NYU MUC-6 System. In MUC-6 [19].
13. L. Kosseim and 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)*, pages 129–140, Sfax, Tunisia, November 1998.
14. L. Kosseim and G. Lapalme. Choosing Rhetorical Structures to Plan Instructional Texts. *to appear in Computational Intelligence*, 16(3), 2000.
15. L. Kosseim, A. Tutin, R. Kittredge, and G. Lapalme. Generating Grammatical and Lexical Anaphora in Assembly Instruction Texts. In G. Adorni and M. Zock, editors, *Trends in Natural Language Generation*, LNAI 103, pages 260–275. Springer, Berlin, 1996.
16. Y. Lallement and M. Fox. Interact: A Staged Approach to Customer Service Automation. In H. Hamilton and Q. Yang, editors, *Canadian AI 2000*, LNAI 1822, pages 164–175, Berlin, 2000. Springler-Verlag.
17. D. Miller, R. Schwartz, R. Weischedel, and R. Stone. Named Entity Extraction from Broadcast News. In *Proceedings of the DARPA Broadcast News Workshop*, Herndon, Virginia, 1999.
18. *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, Baltimore, Maryland, August 1993. Morgan Kaufmann.
19. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, Maryland, November 1995. Morgan Kaufmann.
20. *Proceedings of the Text Retrieval Conference (TREC-8)*, Gaithersburg, Maryland, november 1999.