Coping with Alternate Formulations of Questions and Answers

Brigitte Grau, Olivier Ferret, Martine Hurault-Plantet, Christian Jacquemin, Laura Monceaux, Isabelle Robba, Anne Vilnat

Chapter #

COPING WITH ALTERNATE FORMULATIONS OF QUESTIONS AND ANSWERS

B. Grau, O. Ferret, M. Hurault-Plantet, C. Jacquemin, L. Monceaux, I. Robba, A. Vilnat
LIMSI - CNRS

Abstract: We present in this chapter the QALC system which has participated in the four TREC QA evaluations. We focus here on the problem of linguistic variation in order to be able to relate questions and answers. We present first, variation at the term level which consists in retrieving questions terms in document sentences even if morphologic, syntactic or semantic variations alter them. Our second subject matter concerns variation at the sentence level that we handle as different partial reformulations of questions. Questions are associated with extraction patterns based on the question syntactic type and the object that is under query. We present the whole system thus allowing situating how QALC deals with variation, and different evaluations.

Key words: Terminological variation, extraction pattern

1. INTRODUCTION

The huge quantity of available electronic information leads to a growing need for users to have tools able to be precise and selective. These kinds of tools have to provide answers to requests quickly without requiring users to explore large amount of texts or documents, or to reformulate their request. From this viewpoint, finding an answer consists not only in finding relevant documents but also in extracting relevant parts from them if the question is a factual one, or to summarize them if the request is thematic. This leads us to express the QA problem in terms of an information retrieval problem that can be solved using natural language processing (NLP) approaches.

Pure NLP solutions were studied at the end of the seventies to answer questions as in QUALM, the well-known system of Lehnert (1977). This system analyzed small stories about specific topics (traveling by bus, going to the restaurant, etc.), transformed them into a conceptual representation and answered questions by choosing a strategy depending on the kind of information sought. It consisted of developing reasoning on the conceptual representation making use of general knowledge. In a restricted domain, some recent work such as Extrans (Berri, Mollá Alliod & Hess, 1998) also made use of an NLP approach: its purpose was to analyze the Unix manual, to represent it in a logical form and to make inferences to answer questions. Nevertheless, Extrans proposed to back off to a weaker mode, exploiting keywords when the NLP resolution fails.

The intensive use of semantic and pragmatic knowledge prevents the application of these approaches to open domain questions. As a matter of fact, the resolution strategy has to be adapted to work in such an environment, relaxing the constraints on the conceptual representation. If sentence representations are closer to the surface form, they involve less knowledge and they can be built automatically on a larger scale. Thus, while knowing that the kind of required information remains the same, one can view searching the answer not as an inference problem, but as a reformulation problem: according to what is asked, find one of the different linguistic expressions of the answer in all candidate sentences. The answer phrasing can be considered as an affirmative reformulation of the question, partially or totally, which entails the definition of models that match with sentences containing the answer. According to the different approaches, the kind of model and the matching criteria greatly differ. Strategies range from finding certain words of the questions in the sentence and selecting a noun phrase of the expected type – a minimal strategy applied by all the Question Answering (QA) systems in TREC – to building a structured representation that makes explicit the relations between the words of the question and which is compared to a similar representation of the sentences (Harabagiu, Pasca & Maiorano, 2000; Hovy, Hermjacob & Lin, 2001b). As realizing a complete parse of sentences remains an unsolved problem, our position is halfway. It consists in a partial reformulation of the question, centered on the question focus and expressed by syntactic constraints.

While the expected answer type is rather precise when the questions ask for a named entity — for example the question *When is Bastille Day?* requires a date as answer and the question *What is the name of the managing director of Apricot Computer?* requires a person name — it remains general for other ones, such as questions asking for a definition as in *What is a nematode?* or for a cause. In the former case, the answer type is such as its recognition in sentences can rely on patterns that are independent from the

question terms. Thus, finding an answer mainly requires recognizing an instance of the expected named entity. However, in the latter case, the answer cannot be specified by itself and must be described by a pattern that involves relationships with some question terms and this leads us to talk about linguistic patterns of answers. Nevertheless, whatever criteria are applied, they all require the modeling of linguistic variation at some level.

At the term level, sentences that answer *What is the average salary of a professional baseball player?*, will certainly contain an expression about salary, which might be the *average pay*, and an expression about baseball player, which might be *baseball professional*. The first formulation involves a semantic variation by using a synonym, while the second example relies on a syntactic variation of a noun phrase.

At the sentence level, when looking for a definition, as demanded in *What is epilepsy?*, the answer might be expressed with *epilepsy is a seizure disorder* or *a person has a seizure disorder or epilepsy …*, corresponding to several formulations of the same information involving syntactic variations. These answer formulations can be described by the following patterns: "epilepsy is NP" and "NP or epilepsy" where NP stands for a noun phrase that comprises the answer. The general principle involved in our QA system consists of determining the type of sought information in order to know which patterns better describe an affirmative reformulation. These patterns allow the system to find the answer in a selected sentence.

Before detailing our approach, we will examine in section 2 related work on linguistic variation in order to provide a context. This will be followed in section 3 by a general description of our system, QALC, in order to give a complete vision of our solution and situate within our architecture the role of the different modules we will describe in the further sections. The recognition of term variants, performed by Fastr (Jacquemin, 2001) in our system, help the process that selects relevant passages and the question-sentence pairing process. It will be presented in section 4.

Our criteria for choosing the answering strategy depend on which information is deduced when analyzing the question. It can be one or several of the following features: a) a named entity type that characterizes the answer; b) the syntactic form of the question; c) the question focus, which is a noun that is generally present in the answer formulation; d) the associated answer patterns.

Our question analysis module makes use of a syntactic parser. We will discuss in section 5 why we use such a parser and how it is integrated in our system. We will then discuss how we make use of the different question features. First, the recognition of a noun phrase similar to the question focus in sentences and its impact in the sentence selection process according to other criteria selection will be detailed in section 6. And finally, in section 7,

we will present how question categories lead us to associate with each question a set of reformulation patterns.

2. LINGUISTIC VARIATION IN RELATED WORK

2.1 Paraphrase at the Term Level

Paraphrase is the natural human capacity to use different wordings for expressing the same conceptual content. Many text processing applications have to deal with paraphrase for covering alternate formulations with a similar semantic content. Generating paraphrases is useful in Natural Language Generation (Robin, 1994; Barzilay & McKeown, 2001) because it offers the possibility to use different formulations depending on the context. In Information Retrieval and, especially, in QA applications, it is necessary to cope with paraphrase at various levels of the process. Paraphrase should be accounted for at the indexing level in order to conflate indexes that correspond to similar concepts. Index conflation is taken into consideration by Fastr, which performs term variant recognition (Jacquemin, 2001). At the querying and pairing level, it is also mandatory to recognize variant phrasings in order to associate different formulation of the same information need with its corresponding indexes (Lin & Pantel, 2001).

Once the need for recognizing paraphrases is established, there are several possibilities for processing text documents and conflating paraphrase text chunks. Early attempts in variant conflation such as (Sparck Jones & Tait, 1984) use semantically-rich approaches. In these approaches, it is assumed that (1) a full parse tree can be produced for any sentence in the document and (2) the semantic and morphological links required for the detection of any paraphrase exist in a database. Even though there have been important developments in the design of large scale parsers and in the enrichment of thesauri and term banks, it is unrealistic to pretend that the two preceding requirements can be satisfied in large-scale information access applications such as QA.

Recent developments in large scale paraphrase recognition do not require full in-depth analyses. Instead, they rely on shallow parsers such as Minipar (Berwick, 1991), or a combination of part-of-speech patterns and lexical features (Barzilay & McKeown, 2001; Jacquemin 2001). Although exhaustiveness in paraphrase patterns and associated morphological and semantic links is unrealistic, recent approaches to paraphrase recognition combine machine learning techniques, recycling of human-based semantic or morphological databases, and distributional similarities. In (Barzilay & McKeown, 2001), corpus-based paraphrases are extracted from multiple

translations of the same text through learning algorithms inspired from machine translation techniques. This technique improves upon classical machine translation by providing associations between single and multiple word expressions. With the same purpose in mind, the classical algorithms for extracting semantic classes through distributional similarities were improved by Lin and Pantel (2001) by using similarities between shallow parse trees. The resulting output contains paraphrases at the lexical or phrase level that are missing from manually-generated variations. The approach to paraphrase pattern discovery relies on progressive corpus-based tuning in (Jacquemin, 2001). This approach separates the different levels of variant construction (structural and syntactic, morphological, and semantic). Through corpus-based tuning the structural similarities are extracted. In a second step, the combination of structure and semantic information can be refined by associating specific structures with specific lexical classes based on shallow semantic features (Fabre & Jacquemin, 2000).

In the QA system, QALC, developed at LIMSI, variation is accounted for at the indexing level and at the question analysis level. At the indexing level, variant indexes are conflated through term variant recognition. Term variation involves structural, morphological, and semantic transformations of single or multi-words terms. The semantic links are extracted from WordNet (Fellbaum, 1998) synonymy relations. The morphological links for inflectional morphology result from lemmatization performed by the *TreeTagger* (Schmid, 1999). As for derivational morphology, two words are morphologically related if they share the same derivational root in the CELEX database (CELEX, 1998). Both morphological and semantic links are combined in the structural associations obtained through corpus-based tuning. Term paraphrase recognition is used for dynamic and query-based document ranking at the output of the search engine. Documents that contain variants of the query terms are paired with the corresponding queries. As a result, linguistic variation is explicitly addressed through the exploitation of word paradigms, contrarily to other approaches like the one taken in COPSY (Schwarz, 1988), where an approximate matching technique between the query and the documents implicitly takes it into account.

2.2 Syntactic Variation at the Sentence Level

Paraphrase at the sentence level is tackled when systems have to provide a diversity of texts as a result, as it is the case in narrative or sentence generation in natural language. It also is an issue for systems dealing with this diversity in input, as in the information extraction (IE) field. In a certain way, information extraction is a specialized case of the general QA problem. We will briefly present how the problem was studied in these domains.

2.2.1 Natural Language Generation

Sentence or text generation systems all support the following subtasks (Zock & Sabah, 2002; Hovy, 1996):

- Macroplanning, at the text level, and microplanning, at the sentence level, to determine messages to generate.
- Surface realization, a linguistic component, that orders words, states their lexical categories and manages syntactic constraints. It converts sentence-sized chunks of representation into grammatically correct sentences.

Linguistic variation is addressed in the surface realization task. One of the simplest approaches consists of modeling the sentences to produce by templates when all the different messages have only slight alterations in their linguistic formulation. A template-based approach is generally used when the text to generate is rather fixed, as form letters for example where only a few fields have to be filled. These approaches do not really deal with flexibility of language. A more flexible approach is developed in phrase-based systems that generalize templates. Phrases resemble phrase structure grammar rules, and a pattern describing a sentence is extended for each of its components by using more specific patterns. Such an approach can be powerful and robust. However, it remains difficult to define patterns beyond a certain size in order to avoid incorrect phrase expansion. This complexity does not really constitute a problem when modeling possible formulations of answers, because, in a QA system, patterns have to match existing sentences. Some work like YAG (McRoy, Channarukul & Ali, 2000) and TG/2 (Busemann, 1996) propose a definition language for patterns in order to cover a small sublanguage at different degrees of sophistication. Their formalism deals with canned text (a direct mapping), templates or grammar rules. YAG also includes a general-purpose, template-based grammar for a core fragment of English.

Nevertheless, few generation systems have to deal with the necessity of a great flexibility. The STORYBOOK system (Callaway & Lester, 2001) was conceived to reproduce either the variety or complexity of naturally occurring stories. It performs surface realization with integrated formatting to produce narrative prose similar to the one found in stories written by humans. However, such a system is far too complex with respect to the problem of modeling local variations in sentences. We will see that information extraction systems have abandoned such a general approach for developing specific patterns.

What can be learned from work in the generation domain, and can be reused in QA systems, is the methodology that consists of deciding which conceptual primitives have to be generated, and which patterns to associate

with them to produce a natural realization. For instance, McKeown (McKeown, 1985) defined five ways for providing definitions, found from dictionary and encyclopedia texts: a) identifying the object as a class member (giving the hypernym); b) presenting the constituents of the object (giving the meronyms); c) giving the object attributes; d) using an analogy; e) giving examples.

2.2.2 Information Extraction Field

Information extraction (Gaizauskas & Wilks, 1997) automatically extracts pre-specified sorts of information from natural language texts, typically newswire articles. Information extraction can be viewed as a template filling task. IE grew very rapidly from the late 1980's when DARPA, the US defense agency, funded the MUC program, the major competition in IE. Systems had to extract information about a specified domain and provided instantiated templates. In Muc-4 (ARPA, 1992) for example, the goal was to extract information in the terrorism domain and in MUC-6 (ARPA 96) in the management succession domain.

IE systems are complex, usually consisting of many components. The generic IE system description provided by Hobbs (Hobbs, 1993) allows grasping their main processing stages. Most IE systems perform the following functions:

- Text Zoner: turns a text into a set of text segments.
- Preparser: tries to identify small-scale structures from a sequence of lexical items.
- Parser: its input is a sequence of lexical items, and small-scaled structures (phrases); its output is either fragments of a parse tree or a complete tree.
- Fragment Combiner: tries to turn results of the parser into a semantic representation of a sentence.
- Semantic Interpreter, Coreference Resolution and Template Generator are the last modules, dealing with semantic disambiguation and anaphora resolution at the discourse level before generating the result in a template.

The processes we are interested in are the preparser and the parser. These two steps aim at preparing the semantic interpretation of sentences. Preparser recognizes small-scaled structures that can be recognized with high reliability, as noun groups or verb groups, appositives that can be attached to a noun, as for genitives and "of" prepositional phrases. It also achieves named entity recognition. The preparser usually identifies small-scaled structure by finite-state pattern matching. The parser tries to produce a parse tree of the entire sentence. However, full-sentence parsing of unrestricted texts remains difficult. Thus many IE systems have abandoned it to the benefit of shallow parsing (we will discuss shallow parsing in section 5) or

domain dependant, finite-state pattern matching, trying to locate within the sentence various patterns that are of interest for the application. This break is illustrated by SRI who shifted from the TACITUS system, with its generic text understanding approach, to a dedicated system, FASTUS (Hobbs, Appelt, Bear, Israel, Kameyama, Stickel & Tyson, 1996) that only focuses on elements relevant for the application.

Template filling originated in a project in the mid-60's (Sager, 1981). Templates were defined for extracting information in specialized domains, for instance the medical domain.

A template is defined by the identification of three elements:
– the objects which interacts,
– the relationships representing the interaction between objects,
– the features which are specific to the objects/relationships.

Identification of objects refers to named entity identification. These entities are organizations, person names, currencies, locations, times and dates. Systems in MUC competitions have obtained performances over 90% in this task. QA requires identification of new classes of entities, wider and also more specialized (see (Hovy et al., 2001a) for a detailed hierarchy of answer types).

Interactions between objects can be defined by verb/subject/object relationships in sentences. Features that combine syntactic, semantic and lexical information enable their recognition from sentences. An example of extraction pattern for filling a template would be "<PERSON> was <killed/murdered>".

Extraction patterns were mainly created by experts, even if some work aims at acquiring them automatically, without previous annotations (Riloff, 1996; Yangarber & Grishman, 2000; Poibeau, 2002). However extraction patterns remain very specific to a task, i.e. information sought in a known domain.

The IE task is very close to the answer extraction problem. QA systems have to discover what is searched from questions (equivalent to choosing a template) and specify how to recognize the information (equivalent to the definition of features in terms of lexical, syntactic and semantic criteria for the definition of objects and their relationships), even if the answer to provide (equivalent to a template slot) is more limited.

3. ARCHITECTURE OF QALC

The QALC architecture (see Figure #-1) is quite classical. Its presentation aims at giving a global vision of our solution to the QA problem and positioning the different processes relatively to each other. The analysis

of questions relies on a shallow parser. It intends to extract the following features: a) the syntactic type of the question, b) the question focus, a noun that is likely to be present in sentences that contain the answer, and its noun phrase, c) the answer type, a named-entity tag.

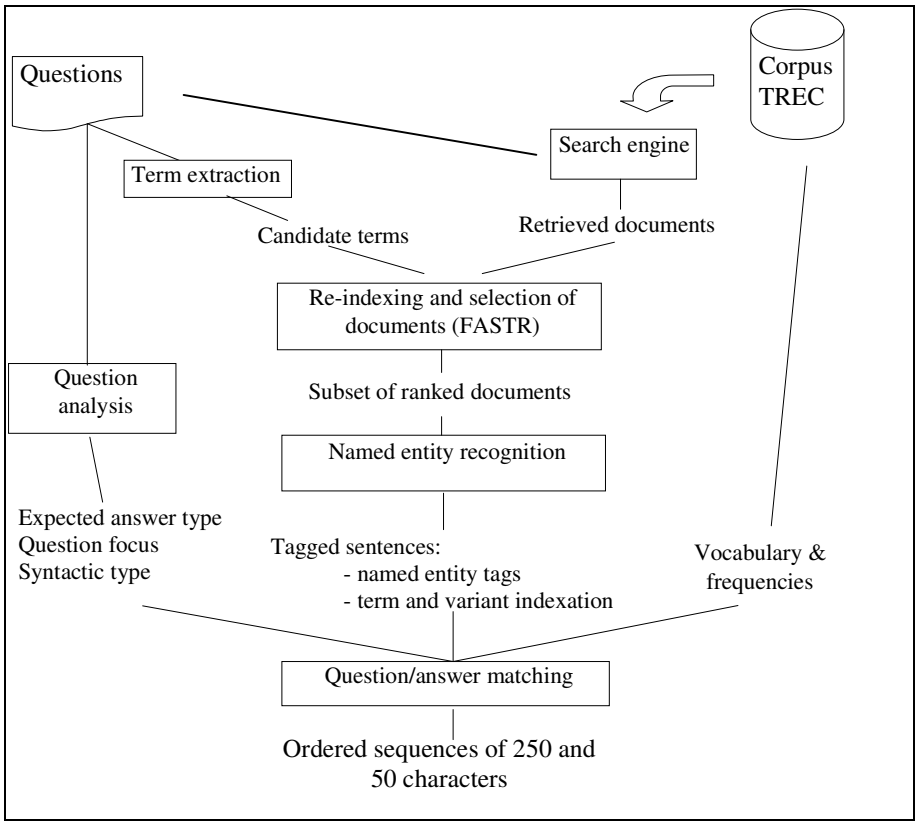


Figure #-1. The QALC system

In order to select the best documents from the results given by the search engine and to locate the answers inside them, we work with terms and their variants, i.e. morphologic, syntactic and semantic equivalent expressions. A term extractor has been developed, based on syntactic patterns that describe complex nominal phrases and their subparts. These terms are used by FASTR (Jacquemin, 1999), a shallow transformational natural language analyzer that recognizes their occurrences and their variants. Each

occurrence or variant constitutes an index that is subsequently used in the processes of document ranking and question/document matching.

Documents are ordered according to a weight computed in function of the number and the quality of the terms they contain. This selection finds its justification during application of the last processes which consist of recognizing named-entities and analyzing each sentence to decide whether it is a possible answer or not. As such processes are time consuming, we attempt to limit their application to a minimal number of documents. Named entities are then recognized in the selected documents.

Finally, the question/answer matching module uses the data extracted from the questions and the documents by the preceding modules for ordering document sentences. Then QALC searches for the precise answer in the selected sentences, using different strategies according to the existence and the nature of the answer type.

4. TERMINOLOGICAL VARIATION

In QALC, terminological variation is used in order to select a sub-set of documents from the results of a search engine. Our selection process will prefer documents where multi-word terms (or variants of them) are near one another from documents containing the same words scattered throughout. Candidate terms are extracted from questions, and QALC can deal with single and multi-word terms.

4.1 Term Extraction

For automatic acquisition of terms from questions, we use a simple technique of filtering through patterns of part-of-speech categories. No statistical ranking is possible because of the small size of the questions from which terms are extracted. First, questions are tagged by the *TreeTagger* (Schmid, 1999). Patterns of syntactic categories are then used to extract terms from the tagged questions. They are very close to those described in (Juteson & Katz 1995), but we do not include post-posed prepositional phrases. The pattern used for extracting terms is:

(((((JJ | NN | NP | VBG)) ? (JJ | NN | NP | VBG) (NP | NN))) | (VBD) | (NN) | (NP) | (CD))

where NN are common nouns, NP proper nouns, JJ adjectives, VBG gerunds, VBD past participles and CD numeral determiners, and the operators are those of regular expression syntax.

The longest string is acquired first and substrings can only be acquired if they do not begin at the same word position as the superstring. For instance,

from the sequence *name*$_{NN}$ *of*$_{IN}$ *the*$_{DT}$ *US*$_{NP}$ *helicopter*$_{NN}$ *pilot*$_{NN}$ *shot*$_{VBD}$ *down*$_{RP}$, the following four terms are acquired: *US helicopter pilot*, *helicopter pilot*, *pilot*, and *shoot*.

The acquisition mode chosen for terms amounts to consider only the substructures that correspond to an attachment of modifiers to the rightmost constituents (the closest one). For instance, the decomposition of *US helicopter pilot* into *helicopter pilot* and *pilot* is equivalent to extracting the sub-constituents of the structure [*US* [*helicopter* [*pilot*]]].

4.2 Variant Recognition through FASTR

The automatic indexing of documents is performed by FASTR (Jacquemin, 1999), a transformational shallow parser for the recognition of term occurrences and variants. Terms are transformed into grammar rules and the single words building these terms are extracted and linked to their morphological and semantic families.

The *morphological family* of a single word *w* is the set *M* (*w*) of terms in the CELEX database (CELEX, 1998) which have the same root morpheme as *w*. For instance, the morphological family of the noun *maker* is made of the nouns *maker*, *make* and *remake*, and the verbs *to make* and *to remake*.

The *semantic family* of a single word *w* is the union *S* (*w*) of the *synsets* of WordNet 1.6 (Fellbaum, 1998) to which *w* belongs. A synset is a set of words that are synonymous for at least one of their meanings. Thus, the semantic family of a word *w* is the set of the words *w'* such that *w'* is considered as a synonym of one of the meanings of *w*. The semantic family of *maker*, obtained from WordNet 1.6, is composed of three nouns: *maker*, *manufacturer*, *shaper* and the semantic family of *car* is *car*, *auto*, *automobile*, *machine*, *motorcar*.

Variant patterns that rely on morphological and semantic families are generated by meta-rules. They are used to extract terms and variants from the document sentences in the TREC corpus. For instance, the following pattern, named NtoSemArg, extracts the occurrence *making many automobiles* as a variant of the term *car maker*:

VM('*maker*') RP? PREP? (ART (NN|NP)? PREP)? ART?(JJ | NN | NP | VBD | VBG)$^{0-3}$ NS('*car*')

where RP are particles, PREP prepositions, ART articles, and VBD, VBG verbs. VM('*maker*') is any verb in the morphological family of the noun *maker* and NS('*car*') is any noun in the semantic family of *car*.

Relying on the above morphological and semantic families, *auto maker*, *auto parts maker*, *car manufacturer*, *make autos*, and *making many*

automobiles are extracted as correct variants of the original term *car maker* through the set of metarules used for the QA-track experiment. Unfortunately, some incorrect variants are extracted as well, such as *make those cuts in auto* produced by the preceding metarule.

4.3 Document Selection

The output of NLP-based indexing is a list of term occurrences composed of a document identifier *d*, a term identifier – a pair *t* (*q,i*) composed of a question number *q* and a unique index *i* –, a text sequence, and a variation identifier *v* (a metarule). For instance, the following index:

LA092690-0038	t(131,1)
making many automobiles	NtoVSemArg

means that the occurrence *making many automobiles* from document *d*=LA092690-0038 is obtained as a variant of term *i*=1 in question *q*=131 (*car maker*) through the variation NtoVSemArg given in Section 4.2.

Each document *d* selected for a question *q* is associated with a weight. The weighting scheme relies on a measure of quality of the different families of variations described by Jacquemin (1999): non-variant occurrences are weighted 3.0, morphological and morpho-syntactic variants are weighted 2.0, and semantic and morpho-syntactico-semantic variants are weighted 1.0.

Since proper names are more reliable clues than common names, each term *t*(*q,i*) receives a weight *P*(*t*(*q,i*)) between 0 and 1.0 corresponding to its proportion of proper names. For instance, *President Cleveland's wife* is weighted 2/3=0.66. Since another factor of reliability is the length of terms, a factor |*t*(*q,i*)| in the weighting formula denotes the number of words in term *t*(*q,i*). The weight $W_q(d)$ of a query *q* in a document *d* is given by the following formula (1). The products of the weightings of each term extracted by the indexer are summed over the indexes *I*(*d*) extracted from document *d* and normalized according to the number of terms |*T*(*q*)| in query *q*.

$$W_q(d) = \sum_{(t(q,i),v)\in I(d)} \frac{w(v)\times(1+2P(t(q,i)))\times|t(q,i)|}{|T(q)|} \qquad (1)$$

For each query *q*, the 200 best ranked documents retrieved by the search engine are processed. Our studies (Ferret, Grau, Hurault-Plantet, Illouz, Jacquemin, 2001) show that 200 is a minimum number allowing that almost all the relevant documents are kept. Mainly two types of weighting curves are observed for the retrieved documents: curves with a plateau and a sharp slope at a given threshold (Figure #-2.a) and curves with a slightly

decreasing weight (Figure #-2.b). Questions in these figures come from the TREC8 data.

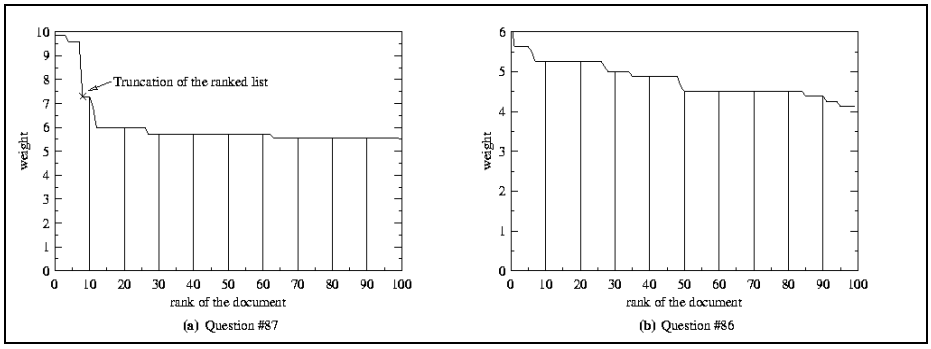


Figure #-2. Two types of weighting curves

The edge of a plateau is detected by examining simultaneously the relative decrease of the slope with respect to the preceding one, and the relative decrease of the value with respect to the preceding one. The following algorithm is used for calculating the cut-off threshold i_0 associated with the weighting scheme W of a given query q:

$$\text{If } \frac{\text{Wq(d2)}}{\text{Wq(d1)}} \le 0.5 \text{ then } \text{i}_0 = 2$$

$$\text{else } \text{i}_0 = \min\left\{ \text{i} \in \{3 \ldots 100\} . \left(\frac{Wq(di) - Wq(di-1)}{Wq(di-1) - Wq(di-2)} \right) \ge 2 \wedge \frac{Wq(di)}{Wq(di-1)} \le 0.8 \right\} \cup \{100\} \quad (2)$$

Through this method, the threshold i_0 is 8 for question 87 (*Who followed Willy Brandt as chancellor of the Federal Republic of Germany?* Figure #-2.a) and 100 for question 86 (*Who won two gold medals in skiing in the Olympic Games in Calgary?* Figure #-2.b). As indicated by Figure #-2.a, there is an important difference of weight between documents 8 and 9. The weight of document 8 is 9.57 while the weight of document 9 is 7.29, because the term *Federal Republic* only exists in document 8. This term has a higher weight because it is composed of two proper names.

Finally, the system retains the i_0 best ranked documents with a minimum number set to 20 and a maximum number set to 100.

4.4 Results and Evaluation

Document selection relies on a quantitative measure, i.e. the document weight, whose computation is based on syntactic and semantic indexes, i.e. the terms and the terminological variants. Those indexes allow the system to take into account words as well as groups of words and their internal relations within the documents. The following examples, issued from selected documents for the TREC9 QA task, show what kind of indexes are added to the question words.

For the question 252, *When was the first flush toilet invented?*, one multi-word extracted term is *flush toilet*. This term is marked by FASTR when recognized in a document, but it is also marked when a variant is found, as for instance *low-flush toilet* in the following document sentence where *low-flush* is recognized as equivalent to *flush*:

> Santa Barbara, Calif., is giving $ 80 to anyone who converts to a *low-flush toilet*.
> 00252.01 flush toilet[JJ][NN] low-flush[flush][JJ] toilet[toilet][NN] 1.00

In all the given examples, after the identification number of the term, the reference term appears, made of the lemmatized form of the words and their syntactic category, followed by the variant found in the sentence, with each word, its lemmatized form and its category, and finally its weight.

In the example above, the term found in the sentence is equivalent to the reference term and thus, its weight is equal to 1.00.

Two multi-word terms extracted from *Who thought of teaching people to tie their shoe laces?*, question 255, are *teach people* and *shoe lace*. In the first example, the morphological variant *teaches* as VB is found for *teaching* as VBG, and *think* as VB for *thought* as VBD.

> You can only say, ' I forgot ' if you have made an effort to notice. Lapp says she *teaches people* to *think* of their mind as a camera.
> 00255.00 think[VBD] think[think][VB] 0.38
> 00255.00 teach people[VBG][NN] teaches[teach][VB] people[people][NN] 0.75

One can notice that we favor variants of multi-word terms that we consider more reliable than variants of single-word terms.

In the second example, the morphological variant *laced* as VBD is found for *lace* as NN, and the syntactic variant *past-participle+name* is found for *name+name*: as a result, *laced shoes* is found for the term *shoe lace*.

> Hepburn, dressed in black slacks, a black turtle neck and sensible *laced shoes*, was swamped with people seeking to have their picture taken.

00255.02 people[NN] people[people][NN] 0.38
00255.03 shoe lace[NN][NN] laced[lace][VBD] shoes[shoe][NN] 0.50

In this case, two variations entail a term weight lower than the weight resulting from a single such variation as in the first example.

The last example shows a semantic variant. *Salary* is a term extracted from the question 337, *What's the average salary of a professional baseball player?*. The semantic variant *pay*, retrieved from WordNet, was recognized in the following sentence:

Did the NBA union opt for the courtroom because its members, whose *average pay* tops $500000 a year, wouldn't stand still for a strike over free agency?
00337.01 salary[NN] pay[pay][NN] 0.25
00337.00 average [JJ] salary[NN] average[average][JJ] pay[pay][NN] 0.40

In order to evaluate the efficiency of the selection process, we proceeded to several measures. We apply our system on the material given for the TREC8 evaluation, one time with the selection process, and another time without this process. At each time, 200 documents were returned by the search engine for each of the 200 questions. When selection was applied, at most 100 documents were selected and subsequently processed by the matching module. Else, the 200 documents were processed. The system scored 0.463 in the first case, and 0.452 in the second case. These results show that the score increases when processing less documents above all because many relevant documents are kept, while less irrelevant documents provide less noise.

The benefit from performing such a selection is also illustrated by the data computed on the TREC9 results, given in Table #-1. We see that the selection process discards a lot of documents for 50% of the questions (340 questions are processed from less than 100 documents). QALC finds the correct answer more often and in a better position for these 340 questions than for the 342 remaining ones. The average number of documents selected, when there are less than 100, is 37. These results are very interesting when applying such time-consuming processes as named entity recognition and question/sentence matching. Document selection will also enable us to apply syntactic and semantic sentence analysis later on.

Table #-1. Evaluation of the ranking process

Number of documents selected by ranking	100	<<100
Distribution among the questions	342 (50%)	340 (50%)
Number of correct answers	175 (51%)	340 (50%)
Number of correct answer at rank 1	88 (50%)	128 (64%)

5. ROBUST PARSERS AND THEIR APPLICATION TO QUESTION-ANSWERING TASKS

As syntactic parsers currently exist, it is natural to use them since some syntactic features are useful to solve our problem. We will thus present the syntactic features suitable for QA. After a general presentation of some parsers based on different approaches, we will illustrate their behavior on some kinds of questions and how we adapted the parser we choose, IFSP (Aït-Mokhtar & Chanod, 1997). Finally, we will detail the QALC feature extraction process from the syntactic analysis of the question.

5.1 Advantages of Syntactic Analysis to QA Application

Question analysis is performed to infer features from questions in order to use them for sentence selection and extraction of potential answer. Basically, question analysis allows:

– the prediction of the type of the answer (for instance, Person, Number…)
– the determination of the important question words, i.e. those that should be present in the answer.

However, simply determining important question words is not always sufficient for finding the right answer: it is sometimes necessary to take into account the relations between these words. In the example shown in Figure #-3, using only the question words leads QALC to find two possible answers. When considering the relations between the words, the answer has to be a person that must be *an astronaut* + *Russian* + *the first to walk in space*. In the second answer, the noun phrase *the first US woman* entails the rejection of this second sentence because of the adjective *US*. Some semantico-pragmatic knowledge might also be available to detect that the two adjectives *Russian* on the one hand and *US* on the other hand, are in contradiction, which would not be the case with *Russian* and *white* or *black.*

Question: *What was the name of the first Russian astronaut to do a spacewalk?*

Answer 1: The broad - shouldered but paunchy Leonov, who in 1965 became **the first man** to walk in space, signed autographs. He drew a dove beside his name for Amanda Clark, 8, of Altadena and pinned on his lapel a Russian-language button from a well-wisher reading . Let us …

Answer 2: Sullivan was **the first US woman** to perform a spacewalk as she and fellow crewman David Leetsma demonstrated satellite refueling …

Figure #-3. Examples of Candidate Answers

A syntactic parse produces a segmentation of the sentence, which contains the syntactic structures (also called constituents or chunks) of the sentence: the noun phrases, the verbal phrases, the prepositional phrases, etc... It also produces the dependency relations that exist between words or constituents. Most often these dependency relations are syntactic relations, like subject, adjective, object… The useful relations between the question terms are expressed in these dependency relations *(*Figure #-4*).*

Question: *What was the name of the first Russian astronaut to do a spacewalk?*

Segmentation: *[SC [NP* What *NP] : v* was *SC] [NP* the name *NP] [PP* of the first Russian astronaut *PP] [IV* to do *IV] [NP* a spacewalk *NP]*?

Dependency relations: SUBJECT (what, was), ADJECTIVE (first, astronaut), ADJECTIVE (Russian, astronaut), etc.

Figure #-4. Example of question syntactic analysis

Once a candidate sentence has been selected, the dependency relations can also be useful for extracting the answer from this candidate. The dependency links between the question phrases give some clues on the function of the equivalent terms in the answer sentence, as we can see it in the following example (Figure #-5) also in (Hovy, Hermjacob & Lin,

2001b). In this example, the question analysis provides the syntactic function of the different noun phrases, the interrogative pronoun *Who* and *Lee Harvey Oswald*, so in the candidate sentences one searches for a person, this person having to be the grammatical subject of the verb *kill* (or a synonym of it), and the object of this verb must be *Lee Harvey Oswald*. So the right answer is *Jack Ruby* and not *J.F Kennedy*.

Question: *Who killed Lee Harvey Oswald?*

Question Syntactic Analysis: Subject (who, kill) Object (kill, Oswald)

→ Answer's Function = Subject

Answer: "*Jack Ruby*, who killed *J.F Kennedy* assassin Lee Harvey Oswald"

Answer Syntactic Analysis: Subject (Ruby, kill), Object (kill Oswald), Noun-Modifier (assassin, Oswald), Noun-Modifier (Kennedy, assassin),

Figure #-5. Utilization of dependency relations for answer extraction

5.2 What kind of Syntactic Parser?

Over the past years, syntactic parsing has seen an important development. Whereas the first syntactic parsers were developed in order to recognize "linguistic phenomena" in ideal sentences, today, syntactic parsers are "realist". They try to parse real-life corpora, which can contain ungrammatical sentences, but also specific phenomena such as dialogs, HTML links, XML tag... The priority of these parsers, called "Robust Parsers", is robustness: a measure of the ability to return a syntactic parse (which can be minimal) independently of the distance between the effective input material and the type of material for which the parser was developed. A Robust Parser is generally determinist, incremental (gradual parse at different specific levels) and based on a corpus. It can recognize minimal structures, but also more complex structures with their dependency relations. Robust parsing relies on heuristics to extract a likely parse of the sentence; some parsers may even give several possible solutions, as it is often the case for prepositional phrase attachment.

It is *a priori* difficult to know which robust parser to use. We can distinguish two families of parsers: the symbolic/linguistic parsers, based on grammatical formalism and the probabilistic/statistical parsers, based on corpus learning (Collins, 1996). At first, we chose to use a linguistic parser for which we could add or modify rules. A probabilistic parser had the

advantage of being corpus-trained but also the drawback that one has to build this corpus, and thus to solve the following problems: how to develop or find an annotated question corpus? how to determine its size? which annotations to use?

At the beginning of their development, the parsers were divided out among three categories, depending on the results they processed:

– constituents-based parsers (SCOL (Abney, 1996) – IPS (Wehrli, 1992)) which return the sentence segmentation. The sentence is first segmented into lexical units and then, into constituents. These parsers are incremental; each module uses the results of preceding module in the processing chain (Figure #-6)

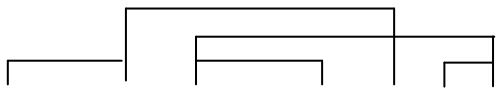
Question: *What was the name of the first Russian astronaut to do a spacewalk?*

Segmentation: [S [INT What] [V was] [NP [DET the] [N name]] [PP of [DET the] [ADJ first] [ADJ Russian] [N astronaut]] [IV to [V do]] [NP [DET a] [N spacewalk]] S] ?

Figure #-6. Example of sentence segmentation

– dependency relations-based parsers (Link Grammar (Sleator & Temperley, 1991)), which return the word (or phrase) dependencies. The Link Grammar parser uses a specific dictionary in which each word corresponds to a formula that specifies the possible links with the other words. The aim of this parser is to find all the links between the sentence words respecting the properties of planarity (no crossing-links) and connectivity (Figure #-7).

Question: *What is the highest dam in U.S?*



Dependency relations: What is the highest dam in U.S?

Figure #-7. Example of dependences way

– constituents and dependency relations based parsers (IFSP (Aït-Mokthar & Chanod, 1997)), which return the sentence segmentation and the phrase dependencies (Figure #-8). These parsers develop a non monotonous reasoning, i.e. some premature decisions could be refined or revisited. Diverse strategies can be used, but generally these parsers search for the minimal sentence segmentation, the simple syntactic relations (subject for instance) and then a greater complexity both in segmentation and syntactic relations.

Question: *Who invented a electric guitar?*

Segmentation: [S [NP Who] :v invented] [NP a electric guitar] ?

Dependency relations: SUBJ(Who, invent), DOBJ(invent, guitar), ADJ(electric, guitar)

Figure #-8. Example of sentence segmentation and syntactic relations

Nowadays, all parsers process the sentence segmentation and the dependency relations, even if the result formats and, above all, the way to obtain these results can be very different.

5.3 Which Robust Parser Use?

5.3.1 Question Processing

The parser we used for TREC-10 is IFSP (see section 5.2). Like most robust parsers, IFSP was developed for large-scale text applications such as information retrieval or terminology extraction. Hence, like the other robust parsers, it is not particularly well suited to parse interrogative forms. This is the reason why the questions processing module we developed had to repair some parsing errors, and this has been done, to a certain extent, by writing specific rules.

In the following examples, we present some forms for which IFSP parsing was erroneous, and we compare the results with the tree other parsers presented in the previous section. It is worth noting that for the sake of time, it was not possible to perform a large-scale comparison on the whole TREC questions corpus[1]. One should also notice that these tests have been

[1] At LIMSI, the new project EASY/EVALDA, on the evaluation of syntactic parsers for French, is under development at the moment. A part of its corpus consists of 500 interrogative forms.

explicitly made on sentences for which IFSP was failing, so it is not surprising that its results compared to those of the other parser were not the best.

– A verb recognized as a noun: in *What year did the Titanic sink?* the four tested parsers (IFSP, FIPS, LGP, SCOL) make the same mistake: *sink* is detected as a noun instead of a verb. In *Why does the moon turn orange?* only the LGP does not make the mistake and recognizes *turn* as the verb.
– The superlative recognized as a noun: in *What metal has the highest melting point?* Only, IFSP, produces an erroneous parse consisting in three noun phrases (one for *highest*, the second for *melting*, and the last for *point*).
– An adjective recognized as a noun: In *Who is the Prime Minister of Canada?* the use of a capital letter for the adjective *Prime* entails an error for IFSP and SCOL, the adjective is not recognized and the noun phrase *Prime Minister* is segmented in two distinct noun phrases (one for *Prime* the other for *Minister*).

Some of the parsing errors can be repaired by writing ad-hoc rules, based on the results given by the parser. Thus, finally, the use of a shallow parser completed by specific rules allowed us to work out a question type module whose results were satisfying (see section 5.4).

Moreover, one attractive aspect in IFSP is that it returns all necessary information, without any need to modify or add any kind of process or knowledge. IFSP returns, on the one hand the constituent segmentation, on the other hand the dependency relationships between the terms, while FIPS, SCOL or LGP return a unique structure, in which the dependency relations are left anonymous. It is possible to obtain from SCOL or LGP the tagging of the dependency relations, but this would imply modifying the grammar.

Later on, several solutions may be adopted to face parser errors. Errors may sometimes be due to the morpho-syntactic tagging. This is the case with the *TreeTagger* used by SCOL, which does not recognize the verb in the two following questions *Why does the moon turn orange? How did Janice Joplin die?*. In SCOL, the morpho-syntactic tagging is not integrated to the parsing, so it is possible to make separate tests and to pinpoint the modules that should be repaired.

Another possibility is to choose a parser for which it is possible and fast to modify or write the syntactic rules necessary to the parsing of interrogative forms. SCOL for instance provides this possibility, so it could be a suitable parser. This is also the case with XIP (Ait-Mokhtar, Chanod & Roux, 2002) developed at Xerox. XIP is a new parser available for French and English, it has better performances than IFSP in processing interrogative forms, and it is possible to modify or to complete its grammar, which was not the case with IFSP.

Conversely some TREC participants chose a parser that could be trained on interrogative forms, those of TREC of course (1300 annotated forms are available) but also some extracted from Internet which represent a large real-life corpus. For Webclopedia, Hovy et al. (2001b) have trained their parser, CONTEXT, on approximately 1150 questions, achieving accuracy of approximately 89 % for both recall and precision.

We could also adopt a whole new approach and work out a dedicated process that would integrate only the kind of parsing useful for the extraction of the information that is specific to the QA task (the question type, the focus… see section 5.4). However, we prefer keeping a generic approach.

5.3.2 Answer Processing

In the system QALC we implemented for TREC10, we used patterns corresponding to local analysis of the candidate answers, instead of a complete parsing of them. These patterns were written using the characteristics obtained from the question analysis module (see section 7). From now on, one of our priorities is to determine whether a robust parser is worth being used for the answer processing.

In opposition to question processing, the parsers have been prepared and even developed for a task such answer processing, indeed robust parsers process large amount of textual data within an acceptable time. These data can contain very complex sentences as well as ungrammatical ones; they can also contain errors due to acquisition, or transcription. It is however worth noting that if the parsing of constituents is most of the time correct, a full parsing processing the relationships between these constituents is rarely obtained.

Despite these difficulties some of the systems participating in the QA track, with a linguistic approach (by opposition to an IR approach), use a syntactic parsing and even sometimes a semantic one.

Hovy and al. (2001b) use CONTEXT a machine-learning based grammar parser whose results are 87.6 % precision and 88.4 % recall. Their system Webclopedia identifies the candidate answer passages and CONTEXT parses their sentences. Then, a matcher module compares the parse trees of the question and of the candidate sentence; but a second and independent matcher uses a word window on the answer text and seems to be useful when the answer parse tree is not complete.

The FALCON system (Harabagiu, Pasca & Maiorano, 2000) uses a statistical parser (Collins, 1996) that has a large coverage on real-word texts. FALCON then transforms the parse tree into a semantic representation that determines the phrase heads and the relations existing between them, but these relations are left anonymous.

5.4 Natural Language Question Analysis in QALC

Question analysis aims at extracting some features from questions that will be used in the answer module. In our system, the question analysis provides several pieces of information to the pairing module:

- a question type with which a list of patterns for extracting the answer will be associated
- an answer type which may be a named entity type or a semantic type to help locating the answer in a sentence
- a question focus to be used as possible criterion for the sentence selection and also to help answer extraction

5.4.1 Question Type

The question type corresponds to the question syntactic form. The detection of a question type gives us a clue to determine the different possible expressions of the answer (see section 7) and will also serve to extract the other question features. After studying the questions of TREC8 and TREC9 along with the sentences containing an answer, we found more than 80 syntactic forms of questions, such as " WhatbeNPofNP ", " HowADJ ", " WhatNPbeNP "…[2]

Other QA systems also determine a question type, at different levels of complexity: for example, the Oracle's system (Alpha, Dixon, Liao, & Yang, 2001) recognizes the wh-words (who, why, where…) to classify the questions. In comparison, (Soubbotin & Soubbotin, 2001) uses 35 more detailed question types. The number of question types generally depends on their utilization in the other modules.

5.4.2 Answer Type

Our question analysis module tries to assign each question an answer type which may be a named entity or a semantic type.

5.4.2.1 Named Entity

The module tries to find an answer type that corresponds to one or several named entity tags sorted by their specificity order, from the most specific to the most general type. The named entity tags are hierarchically organized within semantic classes (see Figure #-9) such as Person, Organization, Location-City, Weight, etc…

[2] NP stands for Noun Phrase in all the following rules

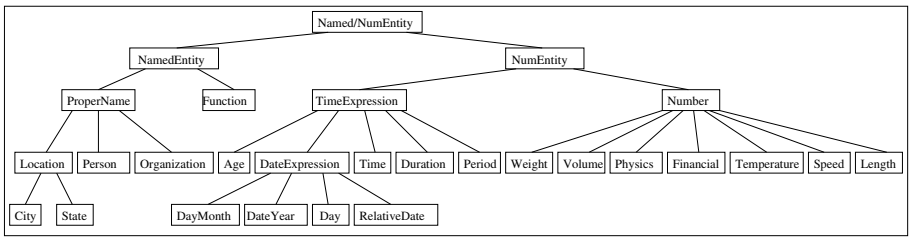


Figure #-9. The hierarchy of tags

To detect a named entity, we built several lexicons: each named entity type is associated with a lexicon that was constructed with the help of WordNet. For example, the LOCATION-CITY lexicon contains, among others, the following terms: capital, town, city, municipality …

The detection of named entity type is performed by different rules, whose conditions are based on the syntactic structure of the question and the semantic classification, for example:

RULE 1: **If** Question Syntactic Form = *WhatBeNP1ofNP2*
and *the head of NP1* belongs to one of EN lexicons
then Named Entity = *EN*

The question *What is the capital of Bahamas?* matches the first condition of RULE 1, with NP1 corresponding to *the capital* and NP2 to *Bahamas*. And the head of NP1, *capital*, belongs to the LOCATION-CITY lexicon, so the answer type of this question is LOCATION-CITY.

As done in QALC, most systems determine the expected answer type by spotting pre-defined patterns in the questions. Prager et al (Prager, Brown, Radev & Czuba, 2000), for instance, had 400 different patterns to identify about 50 types of answers. Using a statistical approach, Ittycherian et al (Ittycheriah, Franz, Zhu & Ratnaparkhi, 2000) based their classification of the answer types on a maximum entropy model. (Clarke et al., 2000) use question categorization and pattern matching heuristics to determine whether a token is a valid candidate term, and thus to select the best scoring answer "snippets". On the other hand, the FALCON system (Harabagiu et al., 2000) used a large knowledge data base, holding a taxonomy of answer types extracted from WordNet hierarchies. Incidentally, question types are often drawn from the classification proposed in QUALM (Lehnert, 1977), one of the first QA systems.

When no named entity can be inferred from the question, the system tries to deduce a more general semantic type.

5.4.2.2 Semantic Type

A semantic type has to belong to a reference knowledge database, the WordNet lexical base for QALC. It means that the answer should be a noun phrase whose head noun is a hyponym of this type. To deduce this type from questions, we wrote rules also based on the question syntactic form, such as:

RULE 2: **If** Question Syntactic Form = *WhatNP*
then Answer Type = *NP-HEAD*

For example, the question What *metal has the highest melting point?* matches the question syntactic form of RULE 2 so the answer type of this question is *metal*. It is generally the noun following the interrogative pronoun.

These different rules were written after studying the questions of TREC8 and TREC9 along with the sentences containing an answer.

The Webclopedia system (Hovy et al., 2001a) also calculates an answer type, which is called "qtarget type", using for this approximately 140 semantic types. However, its hierarchy contains both named entity types and types of answer as definition. As far as we are concerned, we have separated name entity types from definition types since these tow kinds of questions are solved in a very different way.

5.4.3 Focus

A question focus corresponds to a noun phrase that is likely to be present in the answer. For each question, we will determine a focus, the focus head (the head noun of the noun phrase) and the modifiers of the focus head (adjective, complement...) using a set of ordered rules that depend on syntactic and semantic knowledge.

For example, the following rules have been written for questions whose syntactic form is **WhatbeNP1ofNP2**:

RULE 3: **If** NP1-HEAD belong to ABSTRACT lexicon
then **FOCUS = NP2**
FOCUS-HEAD = *NP2-HEAD*

RULE 4: **If** NP1-HEAD belongs to Person or Organization lexicon
then **FOCUS = NP1ofNP2**
HEAD-FOCUS = *NP1-HEAD*

RULE 5: **If** NP1-HEAD belongs to another entity named lexicon

then **FOCUS = NP2**
FOCUS-HEAD = *NP2-HEAD*

Thus, if the question is *What is a group of frogs?* the condition of Rule 3 is satisfied, so the question focus is *frog* and the question focus head is *frog*. On the other hand, the question *Who is the president of United States?* does not match the condition of Rule 3 but the condition of Rule 4, so the question focus is *the president of United States* and the question focus head is *president*.

Generally, the principal criteria to determine the focus are the recognition of the question syntactic form and the type of the main NP. For example, if the question is *What city is the capital of France?,* the question focus is *France* (*capital* belongs to Location lexicon) and not *capital of France*; because the answers, *Paris, France* or *Paris in France* are more frequently encountered than *Paris, capital of France*. Or again in the question *What is the name of the researcher who owned the Calypso?*, the question focus is *researcher*, because *name* is an abstract word and in the answers, *A researcher Cousteau* is more often encountered than *Cousteau, a name of researcher*.

To find if the rule conditions are satisfied, the system uses the results of the robust parser improved with our modifications and also semantic knowledge via the WordNet lexical base. Our system is not the only one to use the notion of focus corresponding to one or several terms of the questions. Soubbotin & Soubbotin (2001), Ittycheriah, Franz & Roukos (2001), Alpha et al. (2001) and Hovy et al. (2001b), all use also this notion. In particular, Hovy identifies the relevant question terms and expands them using WordNet, and Soubbotin & Soubbotin recognize primary words (the words which are indispensable for sentence comprehension). However, the originality of our approach is in the selection of a noun phrase as focus along with the syntactic relations that exist between the focus head and the other question terms.

5.4.4 Results

The rules to find the question focus, the question type and the answer type were written from syntactic criteria provided by IFSP (Aït-Mokthar and Chanod, 1997) and semantic knowledge extracted from the WordNet lexical base. For the TREC10 questions, our question analysis module found about 85 % of correct question focuses, 87 % of correct semantic answer types and 90 % of correct named entity types.

For easy detection of rules that entail errors, it is more convenient to study the results according to question type. For example, the question focus recognition is more difficult for the question type “ WhobeNP ” (for

TREC10, 84 % of correct focus), because the robust parser generates errors that we have to correct. Mistakes can be generated by named entity lexicons[3], which may be incomplete, by robust parser mistakes or by incomplete rules for some question types.

6. QUESTION FOCUS RECOGNITION IN SENTENCES

6.1 Focus Variant Recognition

The question analysis module produces two types of information concerning the focus it has recognized: the head of the focus and a list of modifiers. QALC then tries to recognize this focus in the pre-selected documents. It first detects the head of the focus, and then identifies the noun phrase in which it is enclosed. To determine the delimiters of this noun phrase, we defined a local grammar for the NP in English. This grammar takes into account the tagging made by the *TreeTagger*.

For example, for the question 827:

> *Who is the creator of the Muppets?*
> The focus is *the creator of the Muppets*, with the head: *creator*.

In a document, we find the following NP: <JJ> late <NNS> Muppets <NN> creator <NP> Jim <NP> Henson, which fits the expression: Adjective + Plural Noun + Noun + Proper Noun + Proper Noun.

We also look for NPs with synonyms of the question focus as heads. These synonyms are determined by FASTR. When QALC fails to determine a focus in the question, we decided to consider the proper nouns present in the question as possible focuses since they are also reliable clues for sentence selection. When all these NPs are delimited, we associate them with a score. This score takes into account the origin of the NP and the modifiers found in the question: when the NP contains the modifiers present in the question, its score is increased. The highest score is obtained when all of them are present. In the example of question 827, the score is high since the NP has been obtained directly from the question focus, all the significant words of the focus are present: *creator* and *Muppets*.

[3] A locution lexicon must also be available, to allow the system to resolve ambiguities, for instance the term *span* in *What is the average life span for a chicken?* and the term *expectancy* in *What is the life expectancy of a dollar bill?* are ambiguous while the locution *life span* and *life expectancy* are not and should trigger the search for DURATION entity named type.

When the NP is obtained with a synonym of the head of the focus, the score is slightly decreased, and even more when it is obtained via a proper noun. The scoring algorithm also takes into account the ratio between the number of words present in the question phrase and in the document noun phrase.

For example the score assigned to the NP *their copy of the 13th century Magna Carta* obtained for the question *Which king signed the Magna Carta?,* has a lower score because it has not been obtained from the focus (*king*), but from the proper noun *Carta*, even if it contains all the words of this proper noun phrase, *Magna* and *Carta*.

QALC apply this algorithm for each sentence of the pre-selected documents, thus detecting all the NPs with their associated score. We only keep the NP with the best score in each sentence, which in turn becomes one of the criteria that will be used by the module for sentence selection and the module for answer extraction.

In order to evaluate the relevance of this criterion, we applied this algorithm on all the sentences given as right candidates for the questions of TREC 9. There were 13310 sentences answering to the 693 questions. 57,16 % of them contained an NP similar to the question focus. Overall, at least one focus is found for 89 % of the question collection.

6.2 Sentence Selection

The selection of a set of sentences that may contain the answer to a question is based on the following principle: QALC inspects the selected documents for a question, sentence after sentence, and by means of a function that compares sentences according to their similarity with a question, it constantly keeps in a buffer the N[4] sentences that are the most similar to the question. This comparison relies on the features of the question that have been identified in the sentences of the selected documents by the linguistic modules of QALC. These features are:

– terms;
– focus;
– named entities.

A specific similarity score is computed for each of these features. The score for terms only takes into account single-word terms with no variation. Certainly because of the rough way QALC makes use of terms and the recall level of FASTR, our experiments show that taking multi-word terms or variant terms into account has a negative impact on results. The term score is given by adding the weights of the terms of the question that are in the

[4] N is at least equal to 5. The selected sentences are ranked according to their similarity to the question.

document sentence. The weight of a term integrates its normalized information with regards to a part of the TREC corpus and whether or not it is a proper noun.

The term score is linearly combined with the focus score (see Section 6.1) and the resulting score constitutes the first criterion for comparing two document sentences *S1* and *S2*: if *S1* has a combined score much higher than *S2*[5] ten *S1* is ranked on top of *S2*. Otherwise, the named entity score is used according to the same principle as for terms. It evaluates the extent to which a named entity in a sentence can fit the target of a question when the expected answer is a named entity. This measure relies on the distance between their two types according to the named entity hierarchy of QALC (see section 5.4.2).

When the two preceding criteria are not decisive, the first criterion is used once again but with a smaller threshold for the difference of scores between two sentences. Finally, if there is still an uncertainty, QALC ranks first the sentence that has the shortest matching interval with the question. This interval corresponds to the shortest part of the sentence that gathers all the single-word terms of the question that were recognized in it.

Named Entities play a secondary role in the selection process, as we first select sentences according to the maximum number of terms common with the question, contrarily to (Prager et al., 2000). (Clarke et al., 2000) base their QA system on passage retrieval techniques, rather than on IR techniques and they directly select passages from the whole corpus. Their selection only takes into account the terms extracted from the question. We considered that a sentence with too few terms, even with a named-entity, is not really reliable and we prefer then a NIL answer. As we can see in table 2, section 7.6, our selection process was not performing and we lost a lot of correct sentences.

7. ANSWERS EXTRACTION

When the answer corresponds to a named entity, QALC, as all the TREC systems, mainly relies on this knowledge to find the answer in the selected sentence. Otherwise, QALC exploits the question category found by the question analysis module and its associated extraction patterns.

[5] « Much higher » means that the difference of the scores for *S1* and *S2* is higher than a fixed threshold.

7.1 Named Entity as Answer

Named entities receive a type corresponding to one of the following general category: PERSON, ORGANIZATION, LOCATION (city or place), TIME EXPRESSION (age, date, time, etc.), NUMBER (physic or financial expression, etc.). They are recognized through a combination of lexico-syntactic patterns and significantly large lexical data. The three lists used for lexical lookup are CELEX (CELEX 1998), a lexicon of 160,595 inflected words with associated lemma and syntactic category, a list of 8,070 first names, out of which 6,763 are from the CLR archive (CLR 1998) and a list of 211,587 family names also from the CLR archive.

Contrarily to some systems (for example Hovy et al. 2001a), our categories are rather classic and defined in a way similar as for the MUC task. They all correspond to patterns defined independently from the QA task that can be reused in other contexts.

7.2 Noun or Verb Phrase as Answer

When the expected answer type is not a named entity, the QALC system locates the exact answer within the candidate sentence through syntactic extraction patterns. These patterns include in particular the noun phrase of the question focus head in the candidate sentence. This focus noun phrase has to be as similar as possible to the noun phrase of the focus in the question. The two noun phrases may be exactly the same but, usually, they are variants of each other. They may contain different words but they share at least the focus head.

7.3 Determination of Answer Extraction Patterns

The different extraction patterns were manually determined from corpus analysis. The corpus consisted of the questions and answers provided by the TREC8 and TREC9 evaluation campaigns. From this analysis, we observed that the syntactical structure of a question induces possible syntactical structures for the answer. A typical example of it is the syntactical structure of the direct answer to a question, which is an assertive rewriting of the question. For instance, the answer that our system found for the following TREC10 question is:

Example 1:
Question 1008: *What is the Hawaii's state flower?*
Answer: *Yellow hibiscus is the state flower of Hawaii*

The focus of the question, that has been determined by the rules of the question analysis module (see Section 5.4.3), is *Hawaii's state flower*, and the focus head is *flower*. The noun phrase *the state flower of Hawaii*, in the answer, is a syntactic variant of the focus. The answer of example 1 was extracted from the following candidate sentence: *Yellow hibiscus is the state flower of Hawaii, but Postrzech doesn't recommend them for evening luaus because they close at the end of the day*. The syntactic pattern used to extract this answer is the following:

NPanswer be NPfocus (1)

In this pattern, NPfocus is the noun phrase similar to the question focus within the candidate sentence, and NPanswer is a noun phrase that is supposed to contain the answer. This extraction pattern is very close to a direct answer.

It would be noted that, in this case, we can say *Yellow hibiscus is the state flower of Hawai* as well as *The state flower of Hawaii is the yellow hibiscus*. Thus, the symmetric pattern,

NPfocus be NPanswer (2)

can also be used to extract an answer to the same type of question. This type of question asks for giving the name of the object that is described in the question. There is a relation "object-description", between what is asked for and the focus that describes the object in the question. The possible answer strategies may thus consist of attributing a name to the described object, cf. pattern (2), as well as attributing a description to the named object, cf. pattern (1). It should be noted that, usually, the description contains the answer type. In the example 1, the focus head *flower* is a hypernym of the answer *hibiscus*.

As we just showed, some extraction patterns are closely derived from the syntactic structure of the question. Nevertheless, direct answer structures are not often found in documents, but rather variants of them. For instance, another answer to the question 1008 found in the documents was:

Example 2:
Question 1008: What is the Hawaii's state flower?
Answer: *the state bird, and the Hibiscus the state flower*

This answer is extracted from the sentence *Utah made the allosaurus the state fossil, and Hawaii's Legislature approved making the nene, a native goose, the state bird, and the Hibiscus the state flower*. In this case, the answer (*Hibiscus*) and the focus (*flower*) are in apposition. The answer is a noun which is the object of the implied verb *making*. It means that the description *state flower* is attributed to the flower named *Hibiscus* in accordance with the previous answer examples. Indeed, the relation object-description between the answer and the focus, inferred from the question, may be expressed by different syntactic variants. As a result, the syntactic category of the question, that reflects what is exactly asked for (in our example, the name of a described object), determines some requirements on the answer-focus syntactic structure in the candidate sentences. Therefore, in the construction of extraction patterns from the answer corpus, we were looking for the syntactic structures that could express the different answer-focus relations underlying the questions.

7.4 The Question Categories

We saw in section 5.4 that the questions were parsed in order to get information on the expected answer. A syntactic category is attributed to each question, depending on the syntactic form of the question. However, the same type of question may be expressed by different syntactic forms. For instance, the following questions match a request about a location:

Question 725: *What is the U.S. location of Procter & Gamble corporate offices?*
Question 727: *Where is Procter & Gamble based in the U.S.?*

The phrase *What is the location of* induces a location request, as well as *Where* do. The answer to both questions is a named entity of <LOCATION> type. In both questions, a location relation links an object to the place it is located in, but this relation is expressed through two different syntactic forms, which are two syntactic variants of a same request, and leads to the attribution of the same question category and the same answer type (a named entity type).

Question categorization according to request type is therefore useful for knowing which extraction patterns will be tried. Patterns convey the relation that supports a category, and thus are supposed to be specific to this category. Unfortunately, this is not true for all patterns. Some of them may

pertain to more than one category. For instance, the pattern "NPfocus be NPanswer" is relevant for most of the categories which begin by *What be*. This pattern may express various relations such as a description-object relation (and the symmetric object-description relation), or an object-category relation, or a concept-definition relation.

Each request type may be formulated by one or more syntactic forms. Nevertheless, some general syntactic forms correspond to more than one request type, and therefore have to be refined as soon as the answer extraction patterns differ. For instance, the syntactic form " What be NP? " corresponds to two different request types, on the one hand a request for the definition of a concept, and, on the other hand, a request for the name of a described object. Questions from example 1 are all requests for a definition expressed by the syntactic form " What be NP ? ".

Example 1: Question 912: What is epilepsy?
Answer: a person has a seizure disorder or epilepsy
Pattern: NPanswer or NPfocus

Question 917: What is bipolar disorder?
Answer: manic-depressive illness (also called bipolar disorder)
Pattern: NPanswer (adverb called Npfocus

Question 980: What is amoxicillin?
Answer: antibiotics such as amoxicillin
Pattern: NPanswer such as NPfocus

Question 354: What is a nematode?
Answer: the nematodes are voracious bug killers
Pattern: NPfocus be NPanswer

Patterns associated to each answer show that there is more than one way to answer to a definition question. Conjunction *or*, within the answer to the question 912, indicates that the NPfocus (*epilepsy*) and the NPanswer (*seizure disorder*) have similar meanings[6]. The phrase *(also called*, within the answer to question 917, has more clearly the same role as *or*. In both cases, the concept to be defined is defined by a synonymous phrase. On the other hand, the phrase *such as*, within the answer to question 980, shows a

[6] Obviously, it is not always the case. Conjunction *or* could indicate an alternative in another context.

link between a concept and its hyponym. The concept is then defined by the category to which it belongs. The answer to question 354 shows a third way of answering to a definition request, by the function that the object has. We retrieve here the different possible answers to definition questions exhibited by Mc Keown (1985) (see section 2.2.1). Thus, different types of patterns, that match different types of links between the object to define (the question focus) and its definition (the answer), will be used for extracting the answer.

Example 2 shows the second type of request expressed by the syntactic form " What be GN? ", a request for the named of a described object.

Exemple 2: Question 1213: What is the brightest star?
Answer: Sirius*, the brightest star visible from Earth*.
Pattern: NPanswer , NPfocus

The superlative *brightest* indicates a definite phrase that points out a precise object. The syntactic form that expresses this type of request has then to be refined in " What be definiteNP? " in order to separate it from " What be NP? " which then corresponds to a definition request. Patterns with *also called* and *or* are not relevant for the " What be definiteNP? " category. It should also be noted that the answer is then an hyponym of the focus head *star*.

In QALC, we only make explicit the answer type when it corresponds to a named entity or to a semantic type that will be found in answer sentences. For other types such as definition or object-description, we based in QALC the selection of extraction patterns on the question syntactic category. These two kinds of information lead to different strategies for finding the answer, even if these strategies are complementary: there are different patterns to name a person, according to the characteristics given in the question that belongs to the focus.

7.5 Extraction patterns

The syntactic patterns for answer extraction that we defined from the corpus, always include the noun phrase of the focus and the noun phrase of the answer, which are usually connected by other elements such as comma, quotation marks, a preposition or even a verb. The only exception occurs when the answer is within the noun phrase of the focus. In this case, there is no connecting element between the noun phrase of the focus head and the noun phrase of the answer in the corresponding syntactical pattern. We distinguished three different pattern structures:

(1) NPfocus Connecting-elements NPanswer
(2) NPanswer Connecting-elements NPfocus
(3) NPanswer-within-NPfocus

The answer of example 1 in section 7.3 (*Yellow hibiscus is the state flower of Hawaii)* corresponds to the pattern (2) with the verb *be* as connecting element. The answer of example 2 in the same section (*the state bird, and the Hibiscus the state flower)* corresponds to the pattern (3), since noun phrases in apposition without punctuation mark are processed by our system as a complex noun phrase.

The following example shows another connecting element between the noun phrase of the focus and the noun phrase of the answer in the syntactic extraction pattern. This example comes from our system run on the TREC10 questions.

Example 1: Question 1345: What is the most popular sport in Japan?
Answer: *baseball as the nation's most popular sport*

In this example, the connecting element is the preposition *as*. The answer has been extracted through the pattern " NPanswer as NPfocus " from the candidate sentence *Now, it is threatening to dislodge Japan's stodgy baseball as the nation's most popular sport.* The question focus was *the most popular sport* and the focus head *sport*. We can observe that the focus noun phrase in the answer is a variant of its counterpart in the question. In the candidate sentence, the relation answer-focus is also an object-description relation, the description being attributed to the name of the described sport, *baseball*. Moreover, *sport*, that is the focus head, is a hypernym of the answer *baseball*.

For all the categories we considered 24 extraction patterns. The number of patterns for each question category varies from 2 to 20, with an average of 10 patterns for each category. This approach led us to consider more generic categories than in (Soubbotin & Soubbotin, 2001) and in (Prager et al. 2000), and more generic patterns that are not relative to specific formulations of a kind of request, as all the requests for an acronym, or the requests for a definition for example. Patterns are articulated around a pivot term depending on the question syntactic form, and instantiate some question terms, i.e. the focus terms and eventually the main verb, to produce a partial affirmative form.

The difficulty in finding extraction patterns varies according to the question type. This difficulty is partly due to the small number of some question types within the corpus. For example, there are few *Why* questions (4) and few *How verb* questions (4), such as *Why can't ostriches fly*? (n° 315) and *How did Socrates die?* (n° 198). Moreover, answers to those

questions can hardly be reduced to a pattern. But, for the most part, the difficulty in finding patterns is due to the syntactical diversity of the answers. We hardly found syntactical regularities in the answers to the " What NP be NP? " questions, such as *What format was VHS's main competition?* (n° 426) or *What nationality was Jackson Pollock?* (n° 402) for instance. Depending on the answers, it is the first NP (*format* or *nationality*) or the second NP (*VHS* or *Jackson Pollock*) which plays the main role in the extraction pattern.

Some other QA systems used extraction patterns. Soubbotin & Soubbotin (2001) make patterns the core of their QA system. The indicative patterns they used are different from ours in two ways. First, indicative patterns do not use lemmas and syntactic categories, but sequence and combination of string elements and lists of words and phrases. Secondly, named entities are introduced in patterns. For instance, the pattern " city name ; comma ; country name " is associated to a location question such as *Where is Milan?*. The LCC system (Harabagiu et al., 2001) uses the same type of pattern than those we described, but mainly for definition question type, for which too little information comes from syntactical dependencies. Hovy et al. (2001a) also defined some patterns that, according to its opinion, are too specific to be really useful.

7.6 Evaluation

We carried out an evaluation of the answers that the QALC system retrieved according to whether the question belongs to a category defined by a named entity or not. This evaluation used the TREC10 corpus. First, the evaluation was carried out on the candidate sentences, and then on the 50-characters answers. Our goal was to measure the performance of the two modules processing the answer selection within the retrieved documents. Table 2 sums up the evaluation results of QALC.

Table #-2. Results of QALC

Questions	Number of questions	Number of correct answers (sentences)	Number of correct answers(50-char.)	Correct extraction (sentence->50-char.)
Named entity categories	229	96 (42 %)	82 (36 %)	85 %
Other categories	263	103 (39 %)	63 (24 %)	61 %

Table 2 shows that the QALC system brings better performances concerning the named entity categories, as this type of question usually

includes a number of words which makes easier the selection of the candidate sentences, compared with the definition questions which often include just one word. Moreover, answer extraction is also facilitated by the named entity tagging of the documents.

8. CONCLUSION

Recent work on QA systems has shown that there is a significant difference between systems that operate in restricted domains but at a conceptual level – such as QUALM for instance – and those that come under the bag of words approach that is widely used in Information Retrieval (IR). Between these two types of system, we find the systems that make use of domain-independent knowledge, such as the WordNet database, and rely on general NLP components achieving tasks such as shallow parsing, named entity recognition, term extraction or question analysis. These systems build and process structured representations but unlike conceptual-level ones, their representations are close to the surface form of texts. As a consequence, tackling linguistic diversity, *i.e.* the fact that a concept or an idea may be expressed by many linguistic forms, is one of the important challenges for QA systems nowadays.

In this chapter, we have addressed the part of linguistic diversity that concerns terminological and syntactic variations and we have shown that taking into account these two kinds of variations is actually a mean for increasing the accuracy of a QA system. This observation is congruent with a more general trend emerging from TREC evaluations: the best QA systems are generally those that intensively make use of NLP components in order to bypass linguistic variations. This is one of the important results of the TREC evaluations about QA systems, especially because of the following observation: although the interest of NLP seems to be *a priori* evident for information retrieval tasks, it was never confirmed in an unquestionable way for text retrieval systems (Sparck Jones, 1999), even for basic processing such as morphological analysis (Hull, 1996).

Nevertheless, the reasons for the lack of success of NLP in IR systems concern QA systems too. The lack of robustness of NLP components and the fact that this brittleness is not taken into account by IR systems are two causes that are particularly important. When they work with the Web or with large collections of heterogeneous documents, which is a frequent case of use, IR systems cannot impose restrictions about the texts they process. The results of their possible NLP components are necessarily not perfect and their errors are closely linked to the type of the texts they process. But IR systems generally don't take into account the limits of their components and

they use their results in the same way in all cases. As a consequence, when their overall results contain NLP components, they may be worse than those equipped with less sophisticated components.

The errors of NLP components can be classified into two classic categories: precision errors and recall errors.

Precision errors occur when an entity or a structure that is extracted from a text is not correct, as when a morpho-syntactic tagger sets a noun tag for a word while it is a verb (see Section 5.3.1). Recall errors take place when an entity or a structure that could be extracted from a text is not identified. This is typically the case when a named entity or a term is missed. This kind of error often occurs because a source of knowledge is too sparse. The fact that syntactic parsers generally do not take into account questions (see Section 5.3.1) is another example of this kind of error.

All errors pose real problems but recall errors are the more critical ones. When an entity or a structure is found but is not correct, auto-evaluation mechanisms can be set in some cases for detecting the error. This is more difficult for recall errors as no entity or structure that could trigger a control mechanism exists.

Several strategies were conceived and tested for reducing the effect of these errors and the resulting brittleness of NLP components and of the systems that use them. We distinguish four main strategies. The first twos are local strategies that concern NLP components on their own and the last two are more global:

– A task is achieved by a general NLP component and a set of heuristics corrects its errors in the specific context of its use. This is the approach adopted in QALC for question analysis (see Section 5.3.1): a general syntactic parser allows QALC to identify relations between words and its errors, a part of them coming from a lack of rules for dealing with questions and another part of them resulting from the errors of its tagger, are corrected by a set of specific heuristics that are defined manually. The main drawback of this approach is its lack of generality: heuristics defined for a tool are generally not applicable to another tool achieving the same task. One way to lighten the work that must be achieved for a new tool is to resort to machine learning techniques. Transformation Based Learning (Brill, 1993) is particularly well suitable in that case since it was conceived for learning correction rules.
– A task is performed by a specialized NLP component that is specifically designed for the system in which it is used. This approach is adopted in QALC for the identification of the focus of a question in a document sentence (see Section 6.1). The module that achieves this task in QALC could have relied on a shallow parser for extracting noun phrases from sentences but using a basic set of patterns turned out to be a more flexible

method. More than the previous one, this strategy faces a lack of generality since a specific module must be built for each new context of use.

– A task is achieved in parallel by several modules that rely on different methods. Their results are then combined, generally by a kind of vote. This approach was successfully tested for tasks such as morpho-syntactic tagging (van Halteren, Zavrel & Daelemans, 1998) or speech recognition (Schwenk & Gauvain, 2000). It exploits the fact that several modules based on different methods differ in the errors they make. Thus, minority errors can be discarded by a simple vote. However, it is not clear in which cases this approach can be used. For the moment, it was mainly applied for decision tasks, *i.e.* tasks where the variety of possible errors is limited. It is not sure that it can be successful in a more general context. Moreover, for some kinds of NLP tasks, it is quite heavy to implement and computationally expensive too, even if it is easy to parallelize.
– Several modules that implement different levels of processing are available for achieving a task and the module that is actually used in a specific situation is dynamically chosen according to the characteristics of this situation. One important aspect of this approach is the ability for a system to determine the type of its inputs and the quality of its results. These two evaluations are complementary: the first one is used for selecting the module that is *a priori* the more suited to the current task and the second one is a way to correct this choice if it turns out that it was not correct. The evaluation of inputs is generally taken into account for questions in an accurate way by QA systems (see Section 5.4 for instance). This is not the case for documents, certainly because work about text typology (Biber, 1993) is not mature enough for being applied to QA systems. The evaluation of the results of a system is an active field but it is mainly done in a supervised way: a system is applied on a restricted set of inputs and its results are compared to a reference. On the other hand, little work has been done in this field about the ability of a system to auto-evaluate its results in order to adapt its processing. The FALCON system (Harabagiu et al. 2000) implements this approach for QA: it first retrieves documents with simple requests, evaluate answers extracted from them and build more complex requests (with lexical or semantic expansion for instance) only when these answers are not satisfactory. FALCON distinguishes in this way three levels of processing for requests. In this approach, the brittleness of NLP components is a less important problem, as these components are used only when more basic solutions have already failed. Hence, they can only improve results.

The most widespread strategy of these four is certainly the second one. The last two are the rarer. Their complexity, especially for the last strategy, and their cost from a computational point of view are probably the causes of this rarity of use. Moreover the second point is particularly important for interactive applications or when a high number of requests must be simultaneously processed, as for search engines on Internet.

Nevertheless, the results of a system such as FALCON show that the most promising approach for QA systems, and perhaps more generally for IR systems, consists in using NLP components and above all in controlling their use in order to apply them only when they are likely to improve results. This control can be performed only by developing the ability of QA systems to get information about the situation they face to and the quality of their results, which means achieving a kind of auto-evaluation.

ACKNOWLEDGEMENTS

The authors would like to thank Sasha Blair-Goldensohn for his attentive rereading and Patrick Paroubek for his remarks and for the fruitful discussions which helped us in making things clearer.

REFERENCES

Abney, S. (1996), Partial Parsing via Finite-State Cascades. *Natural Language Engineering*, *2* (4), 337-344.

Aït-Mokhtar, S. & Chanod, J-P. (1997). Incremental finite-state parsing. *Proceedings of Applied Natural Language, Washington, DC.*

Aït-Mokhtar, S., Chanod, J-P. & Roux, C. (2002). Robustness beyond Shallowness: Incremental Deep Parsing. *Natural Language Engineering, special issue on Robust Methods in Analysis of Natural Language Data*, to appear.

Alpha, S., Dixon, P., Liao, C. & Yang, C. (2001). Oracle at TREC 10: Filtering and Question-Answering. *Proceedings of Text retrieval conference, TREC 10, Gaithersburg*, MD. NIST Eds.

Appelt, D., Hobbs, J., Bear, J., Israel, D., Kameyama, M., Kehler, A. et al. (1995) SRI International FASTUS system: MUC-6 test results and analysis. *Proceedings of the 6th Message Understanding Conference (MUC-6), Morgan Kaufman*, pp. 237-248.

ARPA. (1992). Advanced Research Projects Agency. *Proceedings of the Fourth Message Understanding Conference (MUC-4).* San Francisco, California, Morgan Kaufman

ARPA. (1996). Advanced Research Projects Agency. *Proceedings of the Sixth Message Understanding Conference (MUC-6).* San Francisco, California, Morgan Kaufman

Barzilay, R. & McKeown, K. R. (2001). Extracting Paraphrases from a Parallel Corpus. *Proceedings of ACL-EACL'01,* Toulouse.

Berri, J., Mollá Aliod, D. & Hess M. (1998). Extraction automatique de réponses: implémentations du système ExtrAns. *Proceedings of the fifth conference TALN 1998 (Traitement Automatique des Langues Naturelles),* Paris, pp. 12-21.

Berwick, R.C. (1991). Principles of principle-based parsing. In Berwick, R.C., Abney, S.P., & Tenny, C. (Eds.), *Principle-Based Parsing Computation and Psycholinguistics* (pp. 1-38). Kluwer Academic.

Biber, D. (1993). Using Register-Diversified Corpora for General Language Studies, *Computational Linguistics, 19* (2), 219-241.

Brill, E. (1993). *A Corpus Based Approach To Language Learning*, PhD Dissertation, Department of Computer and Information Science, University of Pennsylvania.

Busemann, S. (1996). Best-First Surface Realization, *Proceedings of the 8th International Workshop on Natural Language Generation*, Herstmonceux, Great Britain.

Callaway, C. B. & Lester, J. C. (2001). Narrative Prose Generation. *Proceedings of the 8th IJCAI 2001, Seattle.*

CELEX. (1998). Consortium for Lexical Resources, University of Pennsylvania. From http://www.ldc.upenn.edu/readme_files/celex.readme.html.

Clarke, C.L.A., Cormack,, G.V., Kisman, D.I.E. & Lynam, T.R. , Question answering by passage selection, *Proceedings of the Text retrieval conference, TREC9, Gaithersburg, MD*. NIST Eds.

CLR (1998) Consortium for Lexical Resources, NMSUs, Eds., New Mexico. From http://crl.nmsu.edu/cgi-bin/Tools/CLR/clrcat#D3.

Collins, M. (1996). A New Statistical Parser Based on Bigram Lexical Dependencies. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, ACL-96*, pp.184-191.

Fabre, C.& Jacquemin, C. (2000). Boosting variant recognition with light semantics. *Proceedings of COLING 2000, Luxemburg*, pp. 264-270.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.

Ferret, O., Grau, B., Hurault-Plantet, M., Illouz, G., Jacquemin, C. (2001). *Document selection refinement based on linguistic features for QALC, a question answering system*, Proceedings of the Euroconference on Recent Advances in Natural language Processing (RANLP), Tsigov Chark, Bulgaria.

Gaizauskas, R. & Wilks, Y. (1997). *Information Extraction: Beyond Document Retrieval*. Technical report CS-97-10, Department of Computer Science, University of Sheffield, UK

van Halteren H., Zavrel J. and Daelemans W. (1998) Improving Data Driven Wordclass Tagging by System Combination, *ACL-COLING'98*, pp. 491-497.

Harabagiu, S., Pasca, M., Maiorano, J. (2000). Experiments with Open-Domain Textual Question Answering. *Proceedings of Coling'2000*, *Saarbrucken, Germany*.

Harabagiu, S., Moldovan, D., Pasca, M., Surdeanu, M., Mihalcea, R., Girju, R., Rus, V., Lactusu, F., Morarescu, P., Bunescu, R. (2001) Answering Complex, List and Context Questions with LCC's Question-Answering Server, *Proceedings of the Text retrieval conference, TREC 10, Gaithersburg, MD*. NIST Eds., pp. 355-361.

Hobbs, J.R., (1993). The Generic Information Extraction System. *Proceedings of the Fifth Message Understanding Conference (MUC-5)* (pp. 87-91), Morgan Kaufman.

Hobbs, Appelt, Bear, Israel, Kameyama, Stickel & Tyson (1996). FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text, in Roche and Schabes, eds., *Finite State Devices for Natural Language Processing,* MIT Press, Cambridge MA.

Hovy, E. (1996). Language generation, overview. *In Survey of the State of the Art in Human Language Technology* (chap. 4.1). From http://cslu.cse.ogi.edu/HLTsurvey/HLTsurvey.html

Hovy, E. , Hermjacob, U. & Lin C-Y., Ravichandran, D. (2001a). Towards Semantics-Based Answer Pinpointing, *DARPA Human Technology Conference (HLT)*, San Diego.

Hovy, E. , Hermjacob, U. & Lin C-Y. (2001b). The Use of External Knowledge in Factoid QA. *Proceedings of the Text retrieval conference, TREC 10, Gaithersburg, MD*. NIST Eds. pp. 644-652.

Hull, D. (1996). Stemming algorithms: A case study for detailed evaluation, *Journal of the American Society for Information Science, 47* (1), 70-84.

Ittycheriah, A., Franz, M., Zhu, W-J.& Ratnaparkhi A. (2000), IBM's statistical Question Answering System. *Pre-proceedings of TREC9*, *Gaithersburg, MD*, NIST Eds, pp. 60-65.

Ittycheriah, A., Franz, M. & Roukos, S. (2001). IBM's Statistical Question Answering System – TREC-10. *Proceedings of the Text retrieval conference, TREC 10, Gaithersburg, MD*. NIST Eds.

Jacquemin, C. (1999). Syntagmatic and paradigmatic representations of term variation. *Proceedings of ACL'99*, pp. 341-348.

Jacquemin, C. (2001). *Spotting and Discovering Terms through NLP*. Cambridge, MA: MIT Press.

Justeson, J. & Katz, S. (1995). Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering , 1*, 9-27.

Lehnert, W. (1977). Human and computational question answering. *Cognitive Science* 1, 47-63.

Lin, D. & Pantel, P. (2001). Discovery of Inference Rules for Question-Answering. *Natural Language Engineering 7* (4).

McKeown, K. (1985). Discourse strategies for generating natural-language text, *Artificial Intelligence, 27,* 1-41,

McRoy, S. W., Channarukul, S., & Ali, S. S. (2000). YAG: A Template-Based Generator for Real-Time Systems (System demonstration), *Proceedings of the First International Natural Language Generation Conference (INLG),* Mitze Ramon, Israel.

Poibeau, T. (2002). *Extraction d'information à base de connaissances hybrides.* Thèse de l'université Paris-Nord.

Prager J., Brown, E., Radev, D. & Czuba, K. (2000), One Search Engine or two for Question-Answering, *Proceedings of the Text retrieval conference, TREC9, Gaithersburg, MD*. NIST Eds, pp. 250-254.

Riloff, E. (1996). Automatically Generating Extraction Patterns from Untagged Text, *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, pp. 1044-1049.

Robin, J. (1994). *Revision-based generation of natural language summaries providing historical background: corpus-based analysis, design, implementation and evaluation*. Ph.D. Thesis. CUCS-034-94, Columbia University, Computer Science Department , New York, USA. 357p.

Sager, N. (1981). *Natural Language Information Processing*, Addison-Wesley, Reading.

Schmid, H. (1999). Improvements in Part-of-Speech Tagging with an Application To German. In Armstrong, S., Chuch, K. W., Isabelle, P., Tzoukermann, E. & Yarowski, D. (Eds.), *Natural Language Processing Using Very Large Corpora*, Dordrecht: Kluwer Academic Publisher.

Schwarz, C. (1988). The TINA Project: text content analysis at the Corporate Research Laboratories at Siemens. *Proceedings of Intelligent Multimedia Information Retrieval Systems and Management (RIAO'88)*, Cambridge, MA, pp. 361-368.

Schwenk, H. & Gauvain, J.-L. (2000). *Improved ROVER using Language Model Information*. Proceedings of ISCA ITRW Workshop on Automatic Speech Recognition: Challenges for the new Millenium, pp. 47-52, Paris.

Sleator, D.D.& Temperley, D. (1991). *Parsing English with a Link Grammar*. Technical report CMU-CS-91-196, Carnegie Mellon University, School of Computer Science.

Soubbotin, M. M.& Soubbotin, S. M. (2001). Patterns of Potential Answer Expressions as Clues to the Right Answers. *Proceedings of the Text retrieval conference, TREC 10, Gaithersburg, MD*. NIST Eds.

Spark Jones, K. & Tait, J. (1984). Automatic search term variant generation. *Journal of documentation, 40* (1), 50-66.

Spark Jones, K. (1999). The role of NLP in text retrieval. In T. Strzalkowski (Ed.) *Natural Language Information Retrieval* (pp. 1-24). Boston, MA, Kluwer.

Wehrli, E. (1992). The IPS system. In C. Boitet (Ed.) *Coling-92, GETA*, pp. 870-874.

Yangarber, R. & Grishman, R. (2000). Extraction Pattern Discovery through Corpus Analysis. Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000), Workshop: Information Extraction meets Corpus Linguistics

Zock M & Sabah G. (2002). La génération automatique de textes. In M. Fayol (Ed.) *La production du langage*, Coll. Sciences du Langage, Hermes.