FIDJI: Using Syntax for Validating Answers in Multiple Documents

Véronique Moriceau and Xavier Tannier

Univ. Paris-Sud
LIMSI-CNRS
91403 Orsay, France
{moriceau, xtannier}@limsi.fr

Abstract This article presents FIDJI, a question-answering (QA) system for French. FIDJI combines syntactic information with traditional QA techniques such as named entity recognition and term weighting; it does not require any pre-processing other than classical search engine indexing. Among other uses of syntax, we experiment in this system the validation of answers through different documents, as well as specific techniques for answering different types of questions (*e.g.* yes/no or list questions). We present several experiments which show the benefits of syntactic analysis, as well as multi-document validation. Different types of questions and corpora are tested, and specificities are commented. Links with result aggregation are also discussed.

Keywords: focused retrieval, question answering, syntactic analysis, multi-document validation, result aggregation.

1. Introduction

Document retrieval systems such as search engines provide the user with a set of documents containing relevant information with respect to a query. To obtain a precise answer, the user then needs to locate relevant information within the documents and possibly to combine different pieces of information coming from one or several documents.

To avoid these problems, focused retrieval aims at identifying relevant documents and locating the precise answer to a user question within a document. Question-answering (QA) is a type of focused retrieval: its goal is to provide the user with a precise answer to a natural language question. While information retrieval (IR) methods are mostly numerical and use only little linguistic knowledge, QA often implies deep linguistic processing, large resources and expert rule-based modules.

The classical architecture of a QA system is three-fold. First, the question is analyzed in order to determine the user information need (question focus, keywords, answer type, etc.). Second, a search engine is used to return “candidate documents”, which are expected to contain the answer to the question. Finally, these documents are parsed, and ranked candidate answers are extracted. Whereas the second step is a classical IR process, the first and last steps require using natural language processing techniques, among which the most frequent are POS tagging, syntactic or semantic analysis and named entity recognition.

Since 1999, many evaluation campaigns have been dedicated to question-answering, among them TREC, CLEF, NTCIR or more recently INEX and Quaero. In these campaigns, QA systems have to provide, for each open-domain factoid question, up to five answers ordered by a confidence score. Each answer must be associated with the passage from which it has been extracted (*supporting passage*). State-of-the-art best QA systems obtain close to 70% correct answers for English (Harabagiu and Bejan, 2005), between 55% and 68% for French (Laurent et al., 2005; 2008). The main difficulty is to evaluate what a correct answer is and whether this answer is supported (or validated) by its supporting passage. So far, only the first point is evaluated in classical evaluations. Recently, campaigns such as RTE (Recognising Textual Entailment) or AVE (Answer Validation Exercise) have addressed the second task. Indeed, a correct answer can be considered as relevant by a user only if its supporting passage is relevant and fully validates the extracted answer.

This paper presents FIDJI[1] (Finding In Documents Justifications and Inferences), an open-domain question-answering system for French. Its main goal is to validate answers by checking that all the information given in the question is retrieved in the supporting texts. Our answer validation approach assumes that the different entities of the question can be retrieved, properly connected, either in a sentence, in a passage or in multiple documents. We designed the system so that no particular linguistic-oriented pre-processing is needed.

When a piece of information is being searched, it can be formulated in different ways and some knowledge bases or inferences may be useful to identify it. But, even if lexical databases containing term variations exist (*e.g.* synonyms), conceptual databases for French are not available and, consequently, a semantic approach is not possible. Therefore, our approach consists in extracting and validating answers by using syntactic information, in particular syntactic dependency relations.

In this paper, we present our approach, from question analysis to answer extraction, as well as validation through multiple documents.

We aim at:

- Showing that syntax, used in combination with more “traditional” QA techniques, can lead to interesting results.
- Presenting and evaluating a technique for validating an answer through several passages/documents.
- Testing our approach on two different kinds of corpora, as well as on different kinds of questions.
- Investigating what kinds of questions can benefit from the help of syntax.
- Discussing the links between advanced QA systems and result aggregation.

The remainder of this article presents the general strategy set up in the FIDJI system (Section 2), the syntactic processing (Section 3) and the extraction of the answer (Section 4). Section 5 presents techniques for merging information from several documents to validate answers, and Section 6 describes an evaluation and discusses the results. Finally, Section 7 presents how our approach for answer validation through multiple documents can be a first step towards result aggregation.

[1] This work has been partially financed by OSEO under the Quaero program.

2. FIDJI, a syntax-based strategy

Most question-answering systems can extract the answer to a factoid question when it is explicitly present in texts, but are not able to combine different pieces of information to produce an answer. FIDJI (Finding In Documents Justifications and Inferences), an open-domain QA system for French, aims at going beyond this insufficiency and focuses on introducing text understanding mechanisms.

The objective is to produce answers which are fully validated by a supporting text (or passage) with respect to a given question. The main difficulty is that an answer (or some pieces of information composing an answer) may be validated by several documents. For example:

Q: *Which French Prime Minister committed suicide?*

A: *Pierre Bérégovoy*

P1: *The French Prime Minister Pierre Bérégovoy warned Mr. Clinton against...* P2: *Two years later, Pierre Bérégovoy committed suicide after he was indirectly implicated...*

In this example, the information *French Prime Minister* and *committed suicide* are validated by two different complementary passages. Indeed, this question may be decomposed into two sub-questions, *e.g.* "*Who committed suicide?*" and "*Are they French Prime Minister?*".

Syntactic analysis can provide these kinds of accurate decompositions. Many QA systems use syntactic information, especially dependency relations, mainly for answer extraction. Two approaches emerge: the first one looks for an exact match between the dependency relations of the question and those of the passage (Katz and Lin, 2003), while the second one computes a tree editing distance between the question and the passage (Ligozat, 2007). (Sun et al., 2005), after syntactic and semantic parsing, use the dependency parse tree and the semantic structure of the question for answer extraction.

Some research is also dedicated to question decomposition for QA. Katz et al. (2005) propose a strategy for decomposing questions at a syntactic and semantic level: this allows their QA system START to look for particular pieces of information in multiple resources. When the system cannot find answers to a question, it tries to answer each sub-question. The system uses a number of *parameterized annotations* and semantic templates applied to the whole collection of documents in order to relate questions to information in one or several documents. This system is mainly designed to answer object-property questions (*e.g.* date of birth, a city's population, etc.). FERRET (Hickl et al., 2006), an interactive QA system, performs a syntactic and semantic decomposition of questions which aims at splitting a complex question into a set of semantically simpler questions that the system can answer easily. (Saquete et al., 2004) propose an approach for decomposition of complex temporal questions. (Lin et al., 2008) propose a typology for multi-focus questions which can be decomposed into sub-questions and manually perform their question decomposition strategy to evaluate their approach. Finally, (Hartrumpf, 2008) presents six decomposition classes (temporal, meronymy, etc.), which are employed for annotating German questions and trigger different decomposition methods. Most methods work at the level of semantic representations.

Almost all recent works are based on a syntactic and semantic analysis and often imply a pre-processing of the whole document collection. Our aim is to extract and validate answers by going beyond the exact syntactic match between questions and answers, without using any semantic resources and with as little pre-processing as possible: this is a necessary condition if the system works on large collections such as the Web. In this context of answer validation, the strategy to apply (validation through one or several documents) can be guided by the question, and especially by the expected answer type. Indeed, a lot of factoid questions expect an answer of a specified type. This type can be:

- A named entity type as in "*Who is the president of France?*" which expects an answer of type PERSON;
- A more specific type as in "*Which Russian president attended the G7 meeting in 2007?*" which also expects an answer of type PERSON, but the type is here explicitly specified in the question (*Russian president*).

From now on, we shall call *named entity type* (or *NE type*) and *specific answer type* (or simply *answer type*) these two different notions.

We assume that the answer type can be validated using documents which are different from the document from which the exact answer is extracted.

FIDJI uses syntactic information, especially dependency relations which allow question decomposition. The goal is to match the dependency relations derived from the question and those of a passage and to validate the type of the potential answer in this passage or in another document. Figure 1 presents the architecture of FIDJI.

The document collection is indexed by the search engine Lucene[2]. First, the system submits the keywords of the question to Lucene: the top 100 documents are then processed (syntactic analysis and named entity tagging). Among these documents, FIDJI looks for all sentences containing the highest number of syntactic relations of the question (without limit in the number of sentences). This is the role of module ① in Figure 1. Finally, answers are extracted from these sentences and the answer type, when specified in the question, is validated.

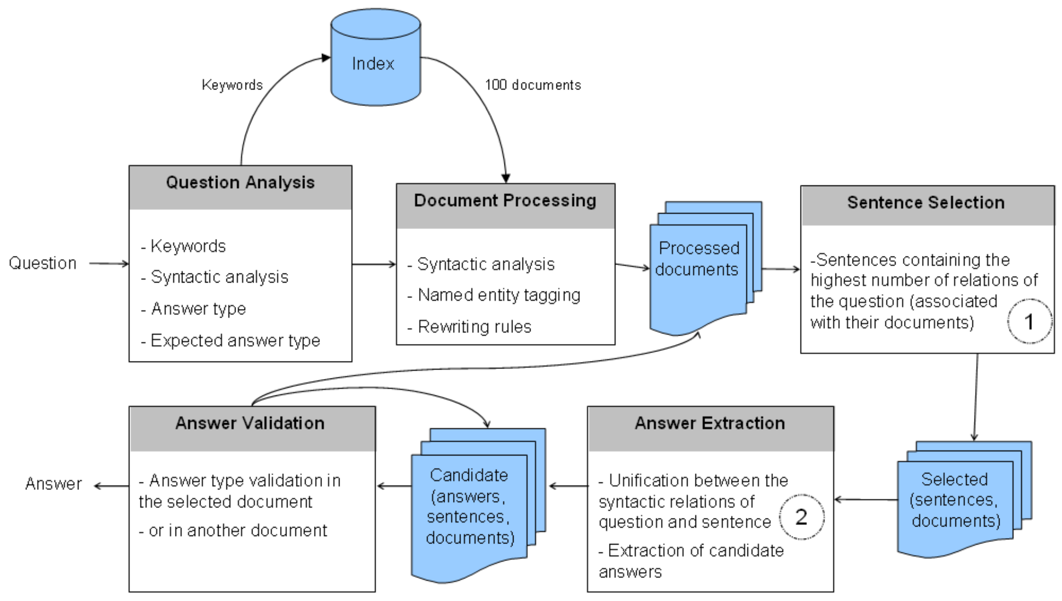
Index
Keywords
100 documents
Question Analysis
- Keywords
- Syntactic analysis
- Answer type
- Expected answer type
Question
Document Processing
- Syntactic analysis
- Named entity tagging
- Rewriting rules
Processed documents
Sentence Selection
-Sentences containing the highest number of relations of the question (associated with their documents)
1
Answer Validation
- Answer type validation in the selected document
- or in another document
Answer
Candidate (answers, sentences, documents)
Answer Extraction
- Unification between the syntactic relations of question and sentence
2
- Extraction of candidate answers
Selected (sentences, documents)

Figure 1. Architecture of FIDJI.

[2] http://lucene.apache.org/java/docs/

3. Syntactic processing

Our approach consists in checking whether all the characteristics of a question (namely the dependency relations) may be retrieved in one or several documents. In this context, FIDJI has to detect syntactic implications between questions and passages containing the answers. Our system relies on syntactic analysis provided by XIP, which is used to parse both the questions and the documents from which answers are extracted.

3.1. Syntactic dependencies and XIP

XIP (Aït-Mokhtar et al., 2002) is a robust parser for French and English which provides dependency relations and named entity recognition. A dependency relation (Mel’čuk, 1984) is an asymmetric binary relation representing a syntactic relationship between two words of a sentence. The first word is the head of the dependency, the second is the modifier. The syntactic structure of a sentence can be represented by dependency relations. Figure 2 shows an example of dependencies produced by XIP’s French grammar on a simple sentence.

Le chat de Pierre mange une souris (literally: *The cat of Peter eats a mouse*)

```
PERSONNE(Pierre)          |  PERSON(Peter)       (NE tagging)
DETERM(chat, le)          |  DETERM(cat, the)
NMOD(chat, Pierre)        |  NMOD(cat, Peter)
PREPOBJ(Pierre, de)       |  PREPOBJ(Peter, of)
SUBJ(manger, chat)        |  SUBJ(eat, cat)
OBJ(manger, souris)       |  OBJ(eat, mouse)
DETERM(souris, un)        |  DETERM(mouse, a)
```

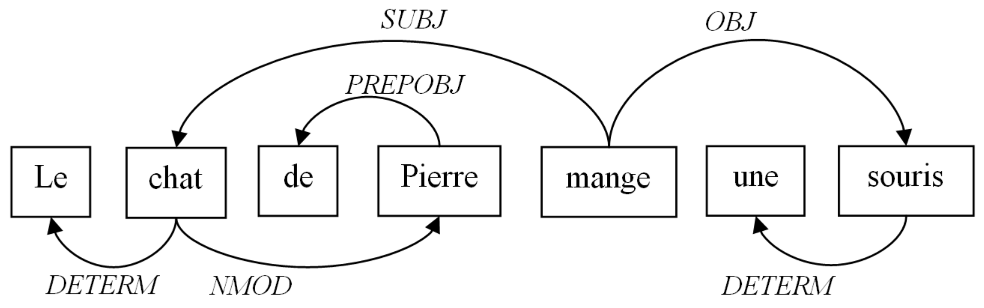

Figure 2. Example of XIP parsing output. The translation in English is given on the right column.

Figure 3 shows the general architecture of XIP’s named entity recognition and syntactic parsing. The dependency relations provided by XIP that are used by FIDJI are mainly: SUBJ (subject), OBJ (object), PREPOBJ (prepositional group), NMOD (noun modifier), VMOD (verb modifier), COORDITEMS (coordinated elements), CONNECT (connector introducing clause).

Named entities (NEs) are tagged using the following classes: person, organization, location, date. XIP’s LIEU (location) can be made more specific (country, region, continent...).

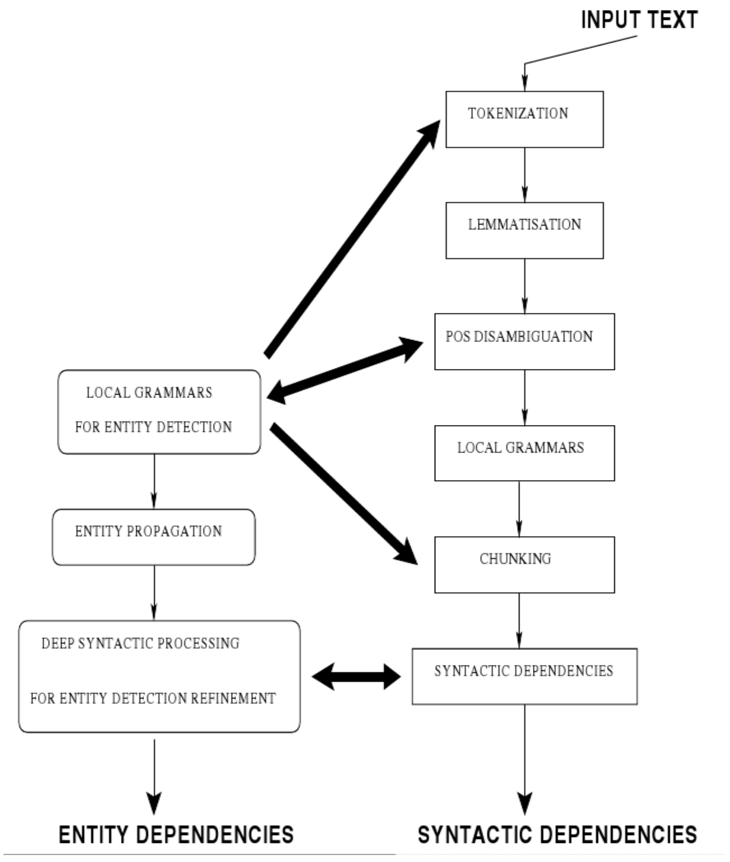

Figure 3. XIP architecture for named entity recognition and syntactic parsing (Brun and Hagège, 2004).

3.2. XIP output enrichment: grammar and named entities

XIP is a product from XRCE (Xerox Research Centre Europe), distributed with encrypted grammars and named entity tagging that cannot be changed by the users. However, it is possible to add resources and grammar rules to the existing ones in order to enrich the representation. This made possible the definition of additional relations/functions to represent some syntactic constructions which are representative of some question types:

- **attributADJ**: for adjectival modifiers as in *le premier homme* (the first man) parsed as *attributADJ(homme, premier)*;
- **attributNN**: for nominal modifiers introduced by a copula verb (*"the first man was Armstrong*" is parsed as *attributNN[copule](man, Armstrong)*).
- **attribut_de**: for genitive complement as in *président de la France* (president of France) parsed as *attribut_de(président, France)*;
- **definition**: definition relation, mostly expressed by appositions as in *"le president Obama"* parsed as *definition(Obama, président)* or *"l'organisation des Nations Unies (ONU)"* parsed as *definition(organisation, ONU)*. More details will be given later about this relation;
- **attribute**: to represent other properties of nouns as in *the president was elected* parsed as *attribute(president, elect)*;

- **answer_type**: to represent the specified answer type (a noun), if explicitly given in the question; we call *simple answer type* this noun, and *extended answer type* the corresponding noun phrase. In our example *"Which Russian president..."*, *president* is the simple answer type, while *Russian president* is the extended answer type.

We also enriched NE tagging with new types: nationality, number, duration, age; and we added features to allow for more precise types. For example, for NUMBER, we added the following features: length, speed, weight, money, physics, so that "*0.55 euro"* in "*a French stamp costs 0.55 euro"* can be tagged as a NE and extracted as an answer to "*What is the price of a French stamp?"*. Other elements are also tagged, such as nouns introducing persons (DECL_PERS): functions (*leader*...), professions (*minister*...), family indications (*father*...).

In this way, NE tagging, combined with question analysis, can help to match the named entity type expected by the question and the extracted answer type.

3.3. Question analysis

Question analysis aims to identify:

- The syntactic dependencies given by XIP and the expected syntactic role of the answer in a declarative sentence corresponding to the question;
- The keywords to be submitted to Lucene: we select all words tagged as noun, verb adjective or adverb by XIP;
- The question type:
    - Factoid (concerning a fact, typically *who*, *when*, *where* questions),
    - Definition (*What is*...),
    - Boolean (expecting a yes/no answer),
    - Complex questions (*why* and *how* questions),
    - List (expecting an answer composed of a list of items).
- The expected type(s): NE type and/or (specific) answer type (see Section 2).

All the items above are determined by using the syntactic structure of the question. The last two items are detailed below.

The question type will guide the strategy that the system will run to extract the answer. NE type and answer type, if existing, are determined once the question type is found.

- Factoid questions will be usually answered by a short noun phrase. They are introduced by a specific *Wh* word or by a specific trigger:
    - Questions in *'qui'* (*'who'*) expect a PERSON or ORGANIZATION NE (*e.g. "Who has been bailed out in 2008?*)[3]. Also, *'Quel ministre'* (*'What minister'*) expects a PERSON. In the latter case, *ministre* is the answer type.
    - Questions in *'quand'* (*'when'*) expect a DATE, as well as *'À quelle date'* (*'At which date'*), etc. Temporal NEs can be made more specific (a year, a day, etc.).

[3] Note that all *"who"* questions are not ambiguous: *"Who is the director of National Intelligence?"* expects only a person, because XIP knows that *director* implies a person name.

    - Questions in *'combien' ('how much/many')* expect a NUMBER, as well as *'Quelle vitesse/température'* (*'what speed/temperature'*), etc., with more specific number NE.
    - Questions in *'où'* (*'where'*) expect a LOCATION, as well as *'À quel endroit'* (*'in which place'*), etc. Location NEs can be made more specific (a region, a country, a continent, a city...), according to XIP's classification.
    - Questions in *'quel'* (*'what/which'*) expect a specific answer type which may not have a corresponding NE type, as *'quelle déclaration'* (*'which declaration'*).
- Definition questions (*'Qu'est-ce que...'* (*'What is'*), *'Que signifie'* (*'What is the meaning of'*), *'Qui est'* (*'Who is'*), ...) will be answered by looking for a 'definition' dependency in texts, with the focus of the question as an argument (see also Section 3.4). A definition question can still have a NE type, as PERSON in *'Who is...'* questions.
- Boolean questions (*'Est-ce que...'*, *'... est-il'*, ...) are processed as described in Section 4.1.3.
- Complex questions are called *'comment'* (*'how'*) and *'pourquoi'* (*'why'*), but can be introduced by other means, such as *'pour quelle raison'* (*'for what reason'*), *'dans quel but'* (*'for what purpose'*), *'de quelle manière'* (*'in which manner'*)... Treatment for complex questions is detailed in Section 4.1.4.
- Questions labelled "list question" are those containing explicitly a plural answer type (*e.g.* "*Which planets...*", "*Who are the...*", etc.). In some cases, the number of items is explicitly specified in the question (*e.g.* "*What are the 7 Wonders of the World?*"). Another difficulty is the identification of implicit list questions: questions such as *"Citez le nom de tous les aéroports de Londres"* (*"Name all the airports of London"*) are easily identified as list questions by the system whereas questions such as *"Qui a découvert la comète Shoemaker-Levy?*" (*"Who discovered the Shoemaker-Levy comet?"*) are identified as factoid questions.

This step is the only one in the entire process that uses some semantic lexical resources (apart from XIP's NE tagging, which is part of the encrypted grammars): a few lists of nouns (about 200) representing persons (as *teacher, minister* or *astronaut*), organizations, locations, nationalities and numerical value units (currencies, physic units...). These lists are very easy to collect and to manage.

The answer to be extracted is represented by a variable ('ANSWER') introduced by our additional grammar rules into dependency relations and named entity tagging. For example, PERSON(ANSWER) means that the answer must have the NE type PERSON; this is determined as explained above. New dependency relations are also created in order to simulate the syntactic role that would have the answer lemma in a declarative sentence having the same form as the question. For example, the relation SUBJ(manger ('*eat*'), qui ('*who*')) in a question is replaced by SUBJ(manger, ANSWER) which means that the answer must be subject of the verb *'manger'.* The slot noted 'ANSWER' is expected to be instantiated by a word, argument of some dependencies of the parsed sentences. This word represents the answer to the question (see Section 4.1).

There are about 250 rules for question analysis. Some examples of rules and analyses are given below. Bold relations are those added by our own grammar rules. On the other hand, some relations are removed from XIP’s regular analysis (strikethrough relations) because they would not be retrieved into a corresponding declarative sentence.

1. Q: Quand fut construite la Tour Eiffel ? (*When was the Eiffel tower built?*)

   Applied rule: `SUBJ(X,Y), CONNECT(X,'quand') → SUBJ(X,Y), DATE(ANSWER)`

   Dependencies (translation in English on the right side):

DATE(ANSWER)	**DATE(ANSWER)**
LIEU(Tour Eiffel)	LOCATION(Eiffel Tower)
SUBJ(construire, Tour Eiffel)	SUBJ(build, Eiffel Tower)
AUX(construire, être)	AUX(build, be)
~~CONNECT(construire, quand)~~	~~CONNECT(build, when)~~

   Question type: factoid

   Expected answer type: DATE (named entity)

2. Q: Quelle déclaration fut adoptée par l'ONU en 1948 ?
   (*Which declaration was adopted by UN in 1948?*)

   Applied rule: `SUBJ(X,Y), OBJ(X,Z), DET_INT(Z,'quel') →`
   `SUBJ(X,Y), OBJ(X,ANSWER), ANSWER_TYPE(ANSWER,Z)`

   Dependencies:

ORGANISATION (ONU)	ORGANISATION (UN)
SUBJ(adopter, ONU)	SUBJ(adopt, UN)
OBJ(adopter, ANSWER)	OBJ(adopt, ANSWER)
VMOD(adopter, 1948)	VMOD(adopt, 1948)
PREPOBJ(1948, en)	PREPOBJ(1948, in)
ANSWER_TYPE(ANSWER, déclaration)	**ANSWER_TYPE(ANSWER, declaration)**

   Question type: factoid
   Expected answer type: déclaration (specific answer type)

3. Q: Qu’est-ce que Linux ? (*What is Linux?*)

   Applied rule: `{PRON['qu’est-ce que'], NOUN(X)} → definition(ANSWER,X)`

   Dependencies:

   definition(ANSWER, linux)

   Question type: definition
   Expected answer type: Ø

4. Q: Qui a inventé le téléphone ? (*Who invented the telephone ?*)

   Applied rule:
   ```
   SUBJ(X, 'qui'), CONNECT(X,'qui') → SUBJ(X,ANSWER),
         PERSON(ANSWER), ORGANISATION(ANSWER), DECL_PERS(ANSWER)
   ```

   Dependencies:

SUBJ(inventer, ANSWER)	**SUBJ(invent, ANSWER)**
OBJ(inventer, téléphone)	OBJ(invent, telephone)
PERSONNE[weight=10](ANSWER)	**PERSON[weight=10](ANSWER)**
ORGANISATION[weight=10](ANSWER)	**ORGANISATION[weight=10](ANSWER)**
DECL_PERS[weight=5](ANSWER)	**DECL_PERS[weight=5](ANSWER)**
~~SUBJ(inventer, qui)~~	~~SUBJ(invent, who)~~
~~CONNECT(inventer, qui)~~	~~CONNECT(invent, who)~~

   Question type: factoid

   Expected answer types: PERSON or ORGANISATION (preferred) or DECL_PERS

As shown by this last example, a single question can be labelled by several expected named entity types, with several weights. This is the case when:

- The question structure is ambiguous: for example, many questions beginning by *"qui"* (*who*) may expect a person or an organisation (*e.g. "Who has been bailed out in 2008?"*).
- Different NEs are acceptable. This is the case when expecting a person and when a name of function or a title can answer the question (*"the Queen of England"* is a good answer to *"Who does not like to be hugged?"*), or when a nationality is as good as a country name.

For that purpose, NE types can be weighted from 1 to 10; this indicates that some types are preferred to others. In practice, NEs are weighted 10, excepted the following three cases:

- Questions expecting a PERSON (weight 10) can also expect a person trigger (DECL_PERS), *i.e.* a NP describing a person (as 'the Queen of England'). This trigger is weighted 5. Definition questions like *'Who is the Queen of England'* do not get a DECL_PERS type, because '*queen*' is a person trigger, so only a real person name will make a proper answer.
- Questions expecting a country name (weight 10) also get a NATIONALITY type (weight 5).
- Questions expecting a typed numerical value (as a length, a speed, a temperature, weighted 10), also get a non-typed numerical value (number only, weight 5).

1 to 10 granularity is not fully used, but has been designed so that more subtle distinctions can be made if necessary. The scoring function (Section 4.3) will take these weights into account in order to rank answers.

3.4. Document processing

To determine if the characteristics of a question can be retrieved in documents, FIDJI detects syntactic implications between a question and documents. For this purpose, the first 100 documents selected by Lucene are syntactically parsed by XIP. Lucene is used with default parameters without stop word removal.

Since information in documents is not always expressed in the same way as in questions, notably because of syntactic variations, reasoning over syntactic dependency relations is essential. As in (Bouma et al., 2007), we have implemented about 40 rewriting rules to account for appositions, coordinations, definitions, etc. Those rewriting rules are appended to the primary grammar. In this way, regardless of the syntactic form of the question, the system is likely to find an equivalent syntactic formulation in the given supporting text: either an exact match between syntactic dependencies of the question and documents or a match between syntactic dependencies of the question and the rewritten dependencies of documents. The following example illustrates the use of a rewriting rule:

Q: A quelle compagnie appartient l'avion détourné par le GIA ?

(*To which company does the plane belong…?*)

```
SUBJ(appartenir, avion)           | SUBJ(belong, plane)
VMOD(appartenir, à)               | VMOD(belong, to)
PREPOBJ(ANSWER, à)                | PREPOBJ(ANSWER, to)
…                                 | …
→ attribut_de(avion, ANSWER)      | → attribut_de(plane, ANSWER)
                                  |                  ("plane of ANSWER")
```

The rewritten dependency relation is obtained by applying the rewriting rule below.

```
SUBJ(appartenir,X), VMOD(appartenir,à), PREPOBJ(Y,à) → attribut_de(X,Y)
```

Now, if the following passage is selected:

Text: L'avion d'Air France, détourné par le GIA en 1994…

(*The plane of Air France, hijacked by the GIA in 1994...*)

```
      attribut_de(avion, Air France)
      …
```

Thanks to the rewriting rules, all the relations of the question match the relations of the text.

The relation 'definition' is also an important extension of the grammar. It has two main interests:

- Answering definition questions: a question like *"What is X ?"* will be answered if the relation 'definition(X, Y)' is found in texts.
- Completing syntactic dependencies when possible. For example, in the sentence *"Le Fonds Monétaire International (FMI) aide la Biélorussie"* (*"International Monetary Fund (IMF) helps Belarus."*), XIP produces the dependency 'SUBJ(aider, FMI)'. Our rules add 'definition(Fonds Monétaire International, FMI)' and the system infers 'SUBJ(aider, Fonds Monétaire International)'.

Definition relation is produced by the detection of the following structures:

- attributNN relation, created by nominal modifiers introduced by a copula verb: *"Le premier homme sur la Lune fut N. Armstrong"* (*"The first man on Moon was N. Armstrong"*) leads to 'definition(man, Armstrong)'.
- Appositions like *"l'organisation des Nations Unies (ONU)"* parsed as 'definition(organisation, ONU)'. Appositions between commas or introduced by a colon as well as *"comme"*, *"tel"* (*"as"*) or more complex expressions as *"du nom de"* or *"nommé"* (*"named"*) are also considered.
- Direct NMOD relation between a singular noun and a proper noun, as *"l'écrivain américain Charles Bukowsky"* (*"the American writer Charles Bukowsky"*), that produces 'definition(écrivain, Bukowsky)'.

4. Finding the answer

Once candidate documents are selected by the search engine and analyzed by the parser, the system compares the document sentences with the result of question analysis, in order to:

- Extract candidate answers;
- Build a supporting passage for each answer;
- Give a score to each answer, so that final answers can be ranked.

4.1. Extracting the answer

The strategy for extracting the answer depends on the type of the question. Syntactic information, at the sentence-level, is used when useful in combination with other types of information.

4.1.1. Factoid questions

Within selected documents, candidate sentences are those containing the highest number of dependencies from the question. Once these sentences are selected, two cases can occur:

1. Question dependencies with an 'ANSWER' slot are found in the sentence. In this case, the lemma instantiating this slot is the head of the answer. The full answer is composed of the head and its basic modifiers (for a noun phrase: noun complements, adjectives, determiners and coordinated elements; for a verbal phrase: verb complements, subject and object). The eventual NE type and answer type of this answer are checked.
   The answer type can be validated by different syntactic relations in the text: definition ("*The French Prime minister, Pierre Bérégovoy"*), attributNN ("*Pierre Bérégovoy is the French Prime minister"*), and sometimes attribut_de (*"la maladie de Parkinson"*, *Parkinson's disease*, literally *"the disease of Parkinson"*). This instantiation of the 'ANSWER' slot is the role of module noted ② in Figure 1.
2. The 'ANSWER' slot does not unify with any word of the passage. In this case, the elements having an appropriate NE type and/or answer type are selected in the sentence. This is done to counterbalance the many parsing errors (or paraphrases). Often, the sentence contains the answer but syntactic dependencies alone do not lead

to it.
When not found in the current sentence, NEs are also selected in the two preceding sentences, but are given a lower score. This technique compensates (very imperfectly) for the absence of a coreference resolution system; it obviously produces some noise, which is generally masked by frequency weighting (the occurrence of a same answer in multiple documents increases its score, see Section 4.3).

4.1.2. List questions

As stated by most evaluation campaigns (TREC, CLEF), we consider presently that items answering list questions should all be found in a single sentence. This is a major limitation and next campaigns of both Quaero and INEX QA tracks should address this issue.

The detection of questions potentially expecting list answers is described in Section 3.3.

List items can be built in two different ways:

- If several candidate answers are found in the same sentence, by any method described in the previous section, then a list is built from these elements.
- If coordination relations concerning some answers are produced by the parser (either through the general grammar or rewriting rules), a list answer is built from all collected elements.

List answers are given a higher weight, but atomic answers are still returned. Indeed, a list question is not necessarily answered only by a list. For example, the question "*Who are the Daltons?"* may be answered by their names, but also by "*4 bandits*".

4.1.3. Yes/no questions

Answering boolean questions is quite close to the task of answer validation (*cf.* the Answer Validation Exercise AVE (Rodrigo et al., 2008)). In this task, participants get a triple {question, candidate answer, supporting text} and have to decide whether the answer to the question is both correct and fully validated by the text.

In 2008, we participated in AVE for French, and ranked first out of 3 for French and second through all languages out of 9 (Moriceau et al., 2008).

We use the same technique when answering yes/no questions: if the rate of question dependencies found in a sentence exceeds a given threshold, the answer is validated, *i.e.* the returned answer is 'yes'. In our system, the threshold was empirically set to 70%.

4.1.4. Complex questions

Complex questions (*"how"*, *"why"*, etc.) do not exist in traditional campaigns. The first Quaero evaluation (Quintard, 2008) introduced a few ones in order to open a new track. We did not set up any specific behavior for these questions. However it is obvious that answers are not short phrases (as for factual questions). For this reason, we chose to provide a full passage as an answer. The three best-scored passages are then returned. On these kinds of questions, the system behaves as a classical passage retrieval system, except that candidate passages are retrieved through syntactic relations instead of keywords only. In a real question-answering situation, these questions would benefit a lot from multi-document aggregation techniques.

4.2. Supporting the answer

Each answer must be supported by a text proving to the user the correctness of this answer. A single supporting passage is provided as follows:

- For each answer, the sentence where this answer got the best score is selected.
- The preceding sentences are also included into the passage (in order to show possible anaphora antecedents) within a limit of 256 characters (classical limit in evaluation campaigns). For complex questions, this limit is extended to 8000 characters, as specified in the Quaero guidelines (Quintard, 2009).

Section 5.4 details the possibility of returning several supporting texts.

4.3. Scoring and ranking answers

FIDJI's scores are not composed of a single value, but of a list of different values and flags. The scoring function has been determined empirically in order to find the best trade-off between different information provided by the different techniques described above.

The figures are listed below, and are presented in decreasing order of importance. This means that for two answers A_1 and A_2, we first examine score s_1. If $s_1(A_1) > s_1(A_2)$, A_1 is ranked better. If $s_1(A_2) > s_1(A_1)$, A_2 is ranked better. If $s_1(A_1) = s_1(A_2)$, then s_2 is considered. In the same way, s_3 is considered only if $s_2(A_1) = s_2(A_2)$, and so on. This is a total order. Note that only scores s3, s4 and s8 are valid for complex questions.

s1. LIST VALUE (only for list questions)
$s1 \in \{0,1,2\}$. As explained in Section 4.1.2, if a question is identified as expecting a list, it is better (but not mandatory) to return a list of answers. Sometimes, the number of expected items is specified in the question ("*What are the 7 Wonders of the World?*"). If the final answer has the right number of items, then $s1 = 2$. If it has not or if this number is unknown, then $s1 = 1$. If the answer is not a list, $s1 = 0$.

s2. NAMED ENTITY VALUE
$s2 \in \{0..10\}$. The question analysis may suggest one or more NE types (see Section 3.3). These types are weighted from 1 to 10, and s2 gets this value if the answer has the corresponding NE type. For example, for the question *"Who said that..."*, possible NE types are PERSON (weight 10), ORGANIZATION (weight 10) and DECL_PERS (person trigger, weight 5). If the answer is *Barack Obama* (PERSON), then $s2 = 10$; if it is *the president of the USA* (DECL_PERS), then $s2 = 5$. If the answer is not an appropriate NE, then $s2 = 0$.

s3. MAJOR KEYWORD RATE
$s3 \in [0, 1]$. Presence of question keywords in the candidate passage is very important. "Major" keywords are proper names and numbers (identified by XIP), as well as the answer type (identified by the question analysis). s3 is the fraction of major keywords from the question that are present in the passage.

s4. QUESTION DEPENDENCY RATE
$s4 \in [0, 1]$. s4 is the fraction of dependencies from the question that are present in the candidate sentence.

s5. ANSWER TYPE VALUE
$s5 \in \{0,1,2\}$. Simple answer type and extended answer type are defined in Section 3.2, and Section 5.1 explains how these types can be validated in a sentence. If the answer

has the appropriate answer type, then *s5 = 1* (*s5 = 2* for the extended answer type). Otherwise, *s5 = 0*.

s6. ANSWER UNIFICATION VALUE
s6 ∈ {0, 1}. If the answer has been found by instantiating the slot 'ANSWER' in question analysis with a specific lemma, then *s6 = 1*. Otherwise, *s6 = 0*.

s7. FREQUENCY WEIGHTING
s7 ∈ [1, ∞[. s7 is the number of times this answer has been found in different passages. See Sections 5.3 and 6.3 for a discussion concerning the use of redundancy in QA.

s8. DOCUMENT RANKING
s8 ∈ {1..100}. Before any linguistic analysis, the search engine (here, Lucene) returns a number of ranked documents in response to a query containing question keywords (see Section 3.4). s8 is the rank given by the search engine to the document containing the answer. In this case, the lower the better (the best document is ranked 1).

s9. DISTANCE VALUE

s9 ∈ R. If several candidate answers are found in the same sentence and if none of them has instantiated the slot 'ANSWER' (*s6 = 0*), it is very difficult to choose between them. As an expedient, we calculate the distance (in words) between the answer and the keywords of the question present in the sentence. Each word of the sentence is numbered from *1* to *n* (*p* is the position of the answer); the barycenter (or centroid) *b* of all question keywords is computed, and s9 is the absolute value of *(p-b)*. Here again, the lower the better.

5. Validating an answer from multiple passages

Many times a candidate sentence does not contain all question dependencies. In some cases, it is possible to check these missing dependencies in other documents or other passages of the same document. In the current state of FIDJI, two validation techniques are used: answer type and answer slot unification.

5.1. Answer type

When existing, answer type is provided by the question analysis (see Section 3.2). If the supporting passage does not specify explicitly that the extracted answer has the correct answer type, this information can be validated in another document. In our example "*Which French minister committed suicide in 1993?*", the answer type is *minister* (while the NE type is PERSON). The *extended* answer type is *French minister*.

Q: *Quel ministre français s'est suicidé en 1993 ?*

Dependencies after rewriting (translation in English on the right side):

SUBJ(se suicider, ANSWER)	SUBJ(commit suicide, ANSWER)
DATE(se suicider, 1993)	DATE(commit suicide, 1993)
answer_type(ANSWER, ministre)	*answer_type*(ANSWER, minister)
attributADJ(ministre, français)	attributADJ(minister, French)

If a text contains the sentence *"Pierre Bérégovoy committed suicide in 1993"*, the slot 'ANSWER' is instantiated by *Pierre Bérégovoy*, which becomes a candidate answer: The first two question dependencies are validated, but the last two ones, concerning the answer type, are missing (was Pierre Bérégovoy a French minister?).

Validation is achieved in two steps. First, the system checks that the candidate answer is a *minister*, through several relations (definition relation and noun modifiers, see Section 3.2). If this is confirmed, the extended answer type is also checked[4].

5.2. Answer slot unification

The other characteristic of the question that can be validated in another passage is the instantiation of the answer slot. In the same way as previously, if a candidate answer is extracted without unifying answer slots in question dependencies, another passage can be found, providing possibly less dependencies, but enabling the instantiation.

5.3. Redundancy

The use of answer redundancy is a very common technique in QA. It simply consists in giving a higher weight to answers that are extracted in several different passages. The idea behind this method is that different ambiguities in the text or the question, combined with imperfections of the extraction algorithm, can lead to bad answers in a specific context. Hopefully, this specific context occurs rarely for a same answer, while good answers should be derived from different ways.

Even if it proved powerful in traditional QA campaigns based on newswire article collections, this technique suffers from several drawbacks:

- Elements (typically, named entities) appearing frequently together with words of the question or with the real answer may benefit from undeserved redundancy bonuses. For example, the question *"What is the real name of Marilyn Monroe?"* can be answered *"J.F. Kennedy"* because this person name appears frequently next to "*Marilyn Monroe"*.
- In less controlled collections (typically, the Web), many documents can contain the exact same text, leading to artificial redundancy.
- In less well-written collections (still the Web) containing spam, tables, lists of keywords or bad language, the gain is not clear.

For all these reasons, redundancy must not be the first criterion for selecting an answer, but it remains an important test for ranking equivalent answers.

[4] And the two relations *ANSWER_TYPE(Bérégovoy, ministre)* and *attributADJ(ministre, français)* are expected to be found in the same sentence.

5.4. Supporting texts

We described in Section 4.2 how a single justification snippet is built. FIDJI is also able to provide multiple supporting texts when necessary. This is the case when answer type or answer slot unification (see above) are validated by different passages. In these particular cases, we know that the first passage is not enough to validate the answer entirely. A new passage is then built from the sentences enabling the validation. Up to three segments of texts can thus support an answer, and we assume that reading all the supports provides all expected information.

For example, CLEF 2006's question 106, *"Quel ancien Premier ministre iranien fut assassiné en 1991 ?"* (*"Which former Iranian Prime minister was killed in 1991"*), leads to the following passage:

> *"... murderers of Chapour Bakhtiar, former Prime Minister, assassinated in 1991 in France, ..."*[5]

This passage is not a full validation of the (correct) answer *Chapour Bakhtiar*, since it does not specify that Chapour Bakhtiar is Iranian. A second text, that does not contain the information concerning the date (1991), but confirms the nationality, is necessary:

> *"... presumed murderers of the former Iranian Prime minister Chapour Bakhtiar ..."*[6]

Answer type validation and answer slot instantiation through multiple passages lead to the presentation of several supporting texts to the user, because all texts carry different information. This is not the case of frequency weighting, where different texts do not introduce different information, but only new ways to write it. This is not useful to the reader and the resulting merging of texts would be redundant.

6. Evaluation

We evaluated our system on corpora and questions from CLEF 2005 (Vallin et al., 2005), CLEF 2006 (Di Nunzio et al., 2006) and Quaero 2008 (Quintard, 2008) test sets:

- The "clean" collections, CLEF05 and CLEF06: composed of about 177,000 well-formed news articles in French from *Le Monde* and *ATS* 1994-1995 (about 2 Gb). These documents are supposed to be syntactically correct. CLEF 2005 questions are factoid and definition questions only, CLEF 2006 introduced a few list questions.
- The "messy" collection, Quaero Q&A Web corpus: a corpus of 2 million Web pages has been collected without quality filtering. This means that any kinds of Web pages (blogs, forums, spam, news, institutions, etc.) can be found, as well as some non-French pages. We obviously expect these documents to be generally much less respectful of French syntactic standards.
  Questions were designed and evaluated by an independent partner. Besides traditional factual and definition questions, lists, yes/no questions and a very few complex questions ('how') have been tested. This is a good opportunity to show that linguistic-oriented techniques can be applied successfully to large, uncontrolled collections.

[5] *" ... les meurtriers présumés de Chapour Bachtiar, ancien premier ministre assassiné en 1991 en France, ...*"

[6] *" ... des assassins présumés de l'ancien premier ministre Chapour Bakhtiar ...*"

In all cases, answers are judged by human assessors and a reference is built. The following aspects have to be evaluated:

- The performance of a syntactic strategy combined with classical QA techniques;
- The performance of the system on different types of corpora;
- The performance of the system on different types of questions;
- The strategy of validating an answer through several passages;
- The actual benefit brought by the different syntactic analysis modules.

6.1. Overall results and different types of questions

Table 1 gives the results that our system obtained on CLEF 2005 (7 participants), CLEF 2006 (7 participants) and Quaero 2008 (3 participants) question sets and document collections. Table 2 provides results by question types and compares them to those obtained by QRISTAL at CLEF 2005 and CLEF 2006 (Laurent et al., 2005; 2006), the best QA system for French for these campaigns. Quaero results are given in Table 3, as well as a comparison with overall results obtained by the best system during the official campaign (Quintard, 2008)[7].

	FIDJI (ranked 1st)	FIDJI (ranked 1-3)
CLEF 2005	66.5%	72.5%
CLEF 2006	50.5%	58.0%
Quaero 2008	28.9%	39.8%

Table 1. FIDJI results on CLEF 2005, CLEF 2006 and Quaero 2008 data. Rate of good answers ranked 1st and 1st to 3rd by the system[8].

	Question type	FIDJI	QRISTAL
CLEF 2005	Factoid (147)	55.1%	59%
	Definition (53)	98.1%	86%
	Total (200)	**66.5%**	**64%**
CLEF 2006	Factoid (152)	46.0%	64%
	Definition (43)	65.1%	83%
	List (5)	60.0%	50%
	Total (200)	**50.5%**	**68%**

Table 2. FIDJI results on CLEF 2005 and CLEF 2006 data, rank 1, by question types. Comparison with QRISTAL results on the same collections.

[7] Results of the official campaign are still anonymous (Quintard, 2008). FIDJI's results presented in this paper were obtained on the exact same corpus and questions. Our factoid/non factoid trade-off differs from (Quintard, 2008), since we added a "definition" class; some definitions are non-factoid and others are so (for example, definition expecting person names, as *"Who is the president of... ?"*).

[8] This measure is common between CLEF and Quaero campaign.

	Question type	FIDJI	Best system
Quaero 2008	Factoid (147)	34.7%	NC
	Definition (56)	26.8%	NC
	Yes/No (19)	31.6%	NC
	List (31)	3.2%	NC
	Complex (3)	33.3%	NC
	Total (256)	**28.9%**	**30.07%**

Table 3. FIDJI results on Quaero 2008 data, rank 1, by question types. Comparison with best system's results on the same collection.

We obtained a lower score on CLEF 2006 data than on CLEF 2005. This can be explained by the introduction of new types of questions for CLEF 2006 (Di Nunzio et al., 2006): definition and list questions. Indeed, for definition questions, new categories (for example, object definitions: *"What is a t-shirt?"*) were added to reduce the number of definition questions which may be answered easily (for example, acronyms (*"What is RKA?"*) and people description (*"Who is Bill Clinton?"*) which are usually found as appositions of proper names). Our scores are lower than those of QRISTAL which also uses a syntactic-based strategy but which benefits from resources such as ontologies and dictionaries.

However, our system would be the second best one as the systems which reached the second position returned 35% of correct answers for CLEF2005 and 46% for CLEF2006.

Quaero results are even much lower. This is the case for all participants and can be explained by several reasons, among which:

- No spam filter has been applied to the corpus, and many questions, especially those concerning known people or events, are polluted by these spam documents.
- Web pages contain structured information (as tables, titles) that can contain very useful information and are not handled by participating QA systems.
- About 10% of the documents (according to an estimation completed by Exalead, personal communication) are not in the appropriate language (*i.e.* French for French evaluation, English for English evaluation).
- Many HTML pages converted from PDF or RTF documents had conversion format problems with special characters, leading to split words.
- Guidelines specified that list answers were to be answered in a single sentence. As all systems, FIDJI considers that sentences are interrupted by string punctuation or new lines. Yet, most list answers from the test set were spread on different list items, separated by carriage returns (*e.g.*, cook recipes).

For Quaero, our system would be the second best one as the system which reached the second position returned 18% of correct answers.

6.2. Evaluation of syntactic modules

Two different syntactic modules can be evaluated separately:

- The selection of candidate sentences through syntactic dependencies present in the question,
- The extraction of candidate answers.

Figure 4 repeats FIDJI's architecture presented in Figure 1 and adds two new modules used for the evaluation.

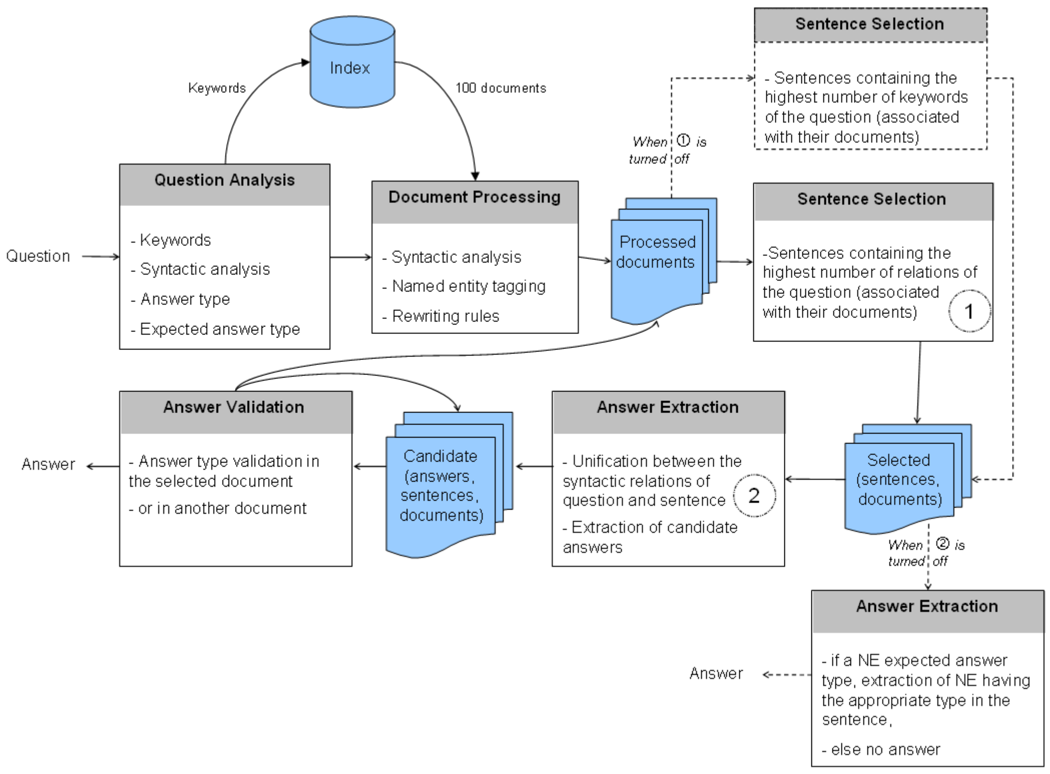
Index
Keywords
100 documents
Question Analysis
- Keywords
- Syntactic analysis
- Answer type
- Expected answer type
Question
Document Processing
- Syntactic analysis
- Named entity tagging
- Rewriting rules
Processed documents
When ① is turned off
Sentence Selection
- Sentences containing the highest number of keywords of the question (associated with their documents)
Sentence Selection
-Sentences containing the highest number of relations of the question (associated with their documents)
1
Answer Validation
- Answer type validation in the selected document
- or in another document
Answer
Candidate (answers, sentences, documents)
Answer Extraction
- Unification between the syntactic relations of question and sentence
2
- Extraction of candidate answers
Selected (sentences, documents)
When ② is turned off
Answer Extraction
- if a NE expected answer type, extraction of NE having the appropriate type in the sentence,
- else no answer
Answer

Figure 4 Architecture of FIDJI for evaluation.

6.2.1. Benefits from using syntactic information for selecting candidate sentences

Most QA systems select candidate passages by checking keywords of the question in texts. FIDJI uses syntactic relations, by selecting sentences containing the most dependencies of the question. In order to evaluate this technique, we compared it with a classical selection of sentences containing a maximum of question keywords (turning off module ① in Figure 4). This means that we still select single sentences, but instead of choosing them according to the number of question dependencies they contain, the system keeps all those that contain the most question keywords (only significant keywords as described in Section 3.3) The evaluation is achieved on the three collections and results are shown in Table 4. In this case, an important distinction should be made between short answers and supporting texts. A supporting text can contain the correct answer, but the finally extracted answer may not be the right one. That is why Table 4 compares both results on correct short answers and passage containing a correct answer.

		Sentence selection with keywords		Sentence selection with dependencies	
		Rank 1	Rank 1-3	Rank 1	Rank 1-3
CLEF 2005	Short answer	57.0%	65.5%	66.5% **(+17%)**	72.5% **(+11%)**
	Passage	63.5%	71.0%	73.0% **(+15%)**	79.0% **(+11%)**
CLEF 2006	Short answer	45.5%	57.5%	50.5% **(+11%)**	58.0% **(+1%)**
	Passage	50.5%	60.5%	53.0% **(+5%)**	62.5% **(+3%)**
Quaero 2008	Short answer	24.2%	31.6%	28.9% **(+19%)**	39.9% **(+26%)**
	Passage[9]	29.1%	34.6%	33.3% **(+15%)**	42.3% **(+22%)**

Table 4. Comparison of FIDJI results with or without selection of candidate sentences with syntactic dependencies. Rate of correct short answers and passages containing correct answers, ranked 1st and 1st to 3rd by the system.

The improvement is substantial (up to 26%). The increase is generally more important for short answer extraction than for passage, and more important for rank 1 answers than for others (with an exception for Quaero collection). It seems to mean that good passages are more numerous, but also that they are more relevant for an effective answer extraction. Selecting candidate passages with the help of syntactic dependencies improves relevant passage ranking.

6.2.2. Benefits from using syntactic information for extracting candidate answers

Evaluating the benefit from using syntax when extracting the answer is done by not applying the technique of 'ANSWER' slot instantiation described at Section 4.1.1 (turning off module ② in Figure 4). Results are presented in Table 5. They show an increase of 47%, 20% and 16% of correct answers at rank 1 when syntactic extraction is present, respectively for collection CLEF 2005, CLEF 2006 and Quaero 2008. This increase is quite important, but it must be noted that syntactic module is the only way to find the answer to some types of questions (especially, definitions). For this reason, it is necessary to distinguish between different types of questions when comparing results.

First, we set apart the questions for which one or more specific named entity types have been identified by the system. Indeed, when module ② is skipped, default behavior is to select a named entity having the appropriate type in the candidate sentence. For the other questions, unplugging this module leads to no answer at all. The global rise depends then strongly on the NE classification and the corresponding question analysis. Among all questions, 151/200 (CLEF 2005), 153/200 (CLEF 2006) and 142/256 (Quaero 2008) have been respectively identified as expecting a specific NE as an answer[10]. It is interesting to remark that even for these questions that benefit from this important clue (the expected NE), syntactic analysis still results in a significant improvement.

[9] For Quaero passage results, yes/no and complex questions have been removed from the pool.

[10] This figure may vary a lot according to systems where NE classification and recognition, as well as question analysis, can be radically different. For example, FIDJI expects a phrase corresponding to a "profession" for the question *"Who is Javier Solana?"*, because it is able to detect these entities.

	Question type	Answer extraction without relations	Answer extraction with syntactic relations
CLEF 2005	NE (151)	57.6%	64.2% (+11.5%)
	Factoid (147)	49.0%	55.1% (+12%)
	Definition (28)	0.0%	96.4% (-)
	Def/pers (25)	72.0 %	100.0% (+39%)
	Total (200)	**45.0%**	**66.5% (+47%)**
CLEF 2006	NE (153)	49.0%	53.6% (+9%)
	Factoid (152)	46.7%	46.0% (-1%)
	Definition (26)	11.5%	50.0% (+4%)
	Def/pers (17)	47.1%	88.2% (+87%)
	List (5)	40.0%	60.0% (+50%)
	Total (200)	**42.0%**	**50.5% (+20%)**
Quaero 2008	NE (142)	35.9%	39.4 (+10%)
	Factoid (147)	34.0%	34.7% (+2%)
	Definition (38)	0.0%	13.2% (-)
	Def/pers (18)	33.3%	55.6% (+67%)
	Yes/No (19)	31.6%	31.6% (+0%)
	List (31)	3.2%	3.2% (+0%)
	Complex (3)	33.3%	33.3% (+0%)
	Total (256)	**25.0%**	**28.9% (+16%)**

Table 5. Comparison of FIDJI results at rank 1 with or without using syntactic relations to extract answers, by question types.

Another distinction should be made between question types. Among definition questions, some concern a person (*"Who is Bill Clinton"* or *"Who is the general secretary of NATO?"*, labeled def/pers in Table 4) while others are general definitions (*"What is a quasar?"*). This last kind falls to 0 correct answer, since no clue other than syntactic relations (or patterns, depending on the system) will make the extraction possible.

Finally, Table 6 produces results obtained by the system without applying any technique of syntactic analysis (turning off both modules ① and ② in Figure 4).

	FIDJI without syntax		FIDJI with syntactic modules	
	Rank 1	Rank 1-3	Rank 1	Rank 1-3
CLEF 2005	40.5%	49.0%	66.5% **(+64%)**	72.5% **(+48%)**
CLEF 2006	40.5%	50.5%	50.5% **(+25%)**	58.0% **(+15%)**
Quaero 2008	21.0%	27.3%	28.9% **(+38%)**	39.8% **(+46%)**

Table 6. Comparison of FIDJI results with all syntactic modules or without using syntax at all. Rate of correct answers ranked 1st and 1st to 3rd by the system.

6.3. Multi-passage validation

In order to evaluate the effect of redundancy weighting on the three collections, we used three different scoring functions:

- f_1: the usual scoring function described in Section 4.3;
- f_2: the same scoring function, by removing *s7* (frequency weighting);
- f_3: a scoring function using *s7* in first position (*i.e.* as the most important criterion).

The results are presented in Table 7. They show that redundancy is useful, and that it allows quite the same increase in CLEF collections as in Quaero corpus. This seems to be in contradiction with other participants' observations, according to which Quaero corpus contained less redundancy than newspapers.

Another conclusion is that redundancy should only be used when other criteria are not discriminant. Using frequency weighting as first criterion leads to a loss of good answers at rank 1.

	Answer ranking with redundancy (f_1)		Answer ranking without redundancy (f_2)		Answer ranking with redundancy first (f_3)	
	Rank 1	Rank 1-3	Rank 1	Rank 1-3	Rank 1	Rank 1-3
CLEF 2005	66.5%	72.5%	62.0%	71.5%	62.0%	70.5%
CLEF 2006	50.5%	58.0%	43.0%	55.5%	47.5%	57.0%
Quaero 2008	28.9%	39.8%	25.0%	35.1%	23.8%	33.4%

Table 7. Comparison of FIDJI results with redundancy normally used in the scoring function, without redundancy and with redundancy as first criterion. Rate of correct answers ranked 1st and 1st to 3rd by the system.

Using several passages (from several documents) to support a single answer is not possible in any currently running evaluation. Multi-passage validation's main purpose is not to provide better short answers, but to give the user a complete proof of the correctness of answers. For these two reasons, it cannot yet be evaluated in an automatic way.

The way these passages are extracted from the collection does not ensure that supporting texts do not carry redundant information, but only that each text brings information that is not present in the other texts. This is not a problem in a QA system, where a text is only used to prove the correctness of an answer to the user, and where a reliable proof is necessary a part of existing, human-made document. But building an aggregated abstract of different supports would be an interesting application for a more polyvalent system (combining for example, IR, QA, aggregated search, automatic abstract, etc.).

6.4. Execution time aspects

The execution strongly depends on the number of documents returned by the search engine, because almost all execution time is spent to parse these documents with XIP. If the maximum number of documents is set to 100 (we used this limit for all results shown in this paper), the parsing lasts 30 seconds per question in average, on a Dual Core 2.66 GHz.

XIP can be used with an API, which allows running several analyses with only one grammar loading, as well as manipulating results with in-built functions. Unfortunately this API is not included in our license, and a lot of time is spent loading grammars and recomposing the results from XIP output file. Using the API would cut execution time in half.

This long execution time is a counterpart of the fact that no pre-processing is necessary to use FIDJI, except the usual search engine indexing. Since the final goal, and especially in Quaero program, is the use of such a system on the entire Web, we think that removing any pre-processing part is mandatory.

6.5. Synthesis

FIDJI has been designed in order to build a high level, realistic question-answering system on the Web. For that purpose, it avoids heavy linguistic pre-processing, that makes currently impossible any scaling to a very large collection of texts. Only a traditional bag-of-word indexing is necessary, and all fine linguistic analysis is performed online on a small subset of documents. Also, no large knowledge base has been used, so that the system is easier to maintain and adaptable to various languages.

We showed that such choices can still lead to good results, since results are quite close to systems appealing to large knowledge and pre-processing (see Section 2). The counterpart is a long execution time, which could be strongly reduced in the future.

The syntactic parser XIP allows to obtain both named entity tagging and syntactic dependencies in a single pass; it is also extensible, making possible a specific analysis for questions as well as rewriting rules for texts.

We also set up in FIDJI a multi-passage validation of answers, through the validation of answer type, the use of redundancy and the gathering of multiple supporting texts. This opens the ways to aggregating results from multiple documents in order to build elaborated answers and justifications.

7. Perspectives: Towards result aggregation

Result aggregation consists in combining information from multiple sources and presenting it efficiently to the user. We propose here some ideas about how result aggregation might enrich FIDJI and QA systems.

7.1. Multi-document answer validation and aggregation

Questions can often be split up into several sub-questions. That is the case for answer type (*"What minister did commit suicide"* is split into *"Who did commit suicide?"* and *"is he/she* *a minister?”*), but also for some other types of structures, as temporal or spatial location, or even verb arguments. The question *“What declaration was adopted by the UNO in 1948?”* can be answered by searching declarations adopted in 1948, and then checking that candidates were really adopted by UNO. The linguistic structure of questions and answers must then be studied very carefully, in order to ensure that no information is lost during the action. The system START (Katz, 1997; Katz et al., 2005) performs these kinds of operations, but with the help of much more resources, especially semantic resources.

FIDJI is currently able to validate answers through several parts of documents, by using answer redundancy or by looking for some missing relations in other passages. Even if FIDJI is not able to produce/generate aggregated results yet, providing the user with a concatenation of several complementary passages which fully validate an answer is a first step towards result aggregation.

7.2. Result aggregation for specific question types

7.2.1. List questions, multiple answer questions

As we already stated, questions expecting an answer composed of several items are of great interest for advanced QA systems, and can benefit a lot from multiple document searching and result aggregation. Current evaluation campaigns artificially restrict list questions to items present in the same sentence. The reason is that traditional QA systems are not designed to merge answers from different sources, and that human assessment would be made quite harder without this restriction.

However, this would correspond to an important user need in several manners:

- Compiling different elements scattered in the collection into a single list of items[11];
- Finding several valid answers to a single question (*“Who is Nicolas Sarkozy?”* leads to “*French president”*, *“former minister”*, *“Carla Bruni’s husband”*, etc.);
- Gathering different answers with different restrictions: temporal (*“Who is the French president?”*: *“Jacques Chirac”* from 1995 to 2007, *“Nicolas Sarkozy”* from 2007), spatial or others.

In all these cases, multiple document searching, natural language generation and result aggregation should become essential to question-answering systems.

As a start, Quaero 2009 campaign will introduce list questions that can be answered by elements found in different elements of the same document.

7.2.2. Complex and opinion questions

List questions are not the only type of questions for which result aggregation and question-answering techniques would be complementary. This is also the case of more complex questions, not very studied so far ((Lee et al., 2008), (Verberne et al., 2007; 2009) for *why* questions, (Yu and Hatzivassiloglou, 2003) for opinion questions). Questions concerning procedures (in short, *“how”* questions), reason explanation (*“why”*) or opinions can hardly find a complete answer in a single part of document. The usual “short answer” QA model is

[11] Note that list questions can be specified in an explicit (*“What are the different words to express the snow in Inuit?”*) or implicit way (*“Who got a golden parachute in 2008?”*).

no longer valid for answering these questions, but it obviously stays part of the QA domain, particularly in the question analysis part.

For example, concerning opinion questions, the system should be able to locate opinions in documents, their polarity, their holder, their strength, etc., and to produce or generate a synthetic "answer" in a suitable way, possibly in a table as what (Lin and Liu, 2008) propose for multi-focus factoid questions.

8. Conclusion

In this article, we showed how the use of syntactic analysis can help a question-answering system to produce good results. FIDJI's performances, on CLEF collections as well as on Quaero QA campaign, are close to best state-of-the-art systems, without having recourse to large external resources used by these systems. Also, FIDJI does not require any pre-processing other than a classical search engine indexing. This means that it can be plugged to any search engine and be used on a real Web application. On the contrary, scalability is a big problem for systems using heavy resources. The counterpart of this advantage is a poor execution time, but we hope to reduce this issue.

We presented experiments measuring the benefit brought by the use of syntactic modules; we also studied the behavior of our system and its syntactic modules according to the different types of questions, as well as different corpora (*i.e.* newspaper articles *vs.* Web collection). In particular, we showed that using syntactic analysis on Web pages, obviously less "clean" (bad spelling, bad syntax, spam), could lead to interesting results. FIDJI's performances decreased on such a corpus but not more than other participants'.

In a candidate sentence, missing relations are often due to the fact that the meaning behind these relations is expressed by a paraphrase. In this case, a linguistic validation is necessary[12]. Some techniques for finding lists of paraphrases out of context exist (Lin and Pantel, 2001; Barzilay and Lee, 2003; Harabagiu and Hickl, 2006; Fujita and Sato, 2008); but in our case, one of our perspectives is to use the lexical and syntactic contexts, as well as the fact that other similar documents have been retrieved by the search engine, to prove that two sets of dependencies are paraphrases of each other.

We also presented some approaches used to validate an answer through several documents, in order to show a complete proof of the correctness of the answer to the user. This opens the way to many perspectives in multi-document search as well as in result aggregation.

New evaluation campaigns, as INEX or Quaero, initiate new types of questions, new corpora, new evaluation techniques and new bridges with other fields, in order to use QA for answering more user needs. We aim at adapting FIDJI on English language and working on the issues raised by these new challenges, like list answer building and complex question answering.

[12] For example, a question *"Who plans to acquire Yahoo?"* can be answered by *"Microsoft said it wants to buy Yahoo"*. We need to validate that "want to buy" is equivalent to "plan to acquire".

References

Aït-Mokhtar, S., Chanod, J.-P. & Roux C. (2002). Robustness beyond shallowness: incremental deep parsing. *Natural Language Engineering, 8 (2/3)* (pp. 121-144). Cambridge University Press..

Barzilay, R., & Lee, L. (2003). Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. *in Proceedings of HLT-NAACL 2003*, (pp. 16-23). Edmonton, AB, Canada.

Bouma, G., Fahmi, I., Mur, J., van Noord, G., van der Plas, L., & Tiedemann, J. (2007). Linguistic knowledge and question answering. *Traitement Automatique des Langues , 46* (3), 15-39.

Brun, C., & Hagège, C. (2004). Intertwining deep syntactic processing and named entity detection. *Proceedings of España for Natural Language Processing (EsTAL 2004).* Alicante, Spain.

Di Nunzio, G. M., Ferro, N., Mandl, T., & Peters, C. (2006). CLEF 2006: Ad Hoc Track Overview. *Working Notes for the CLEF 2006 Workshop.* Alicante, Spain.

Fujita, A., & Sato, S. (2008). Computing Paraphrasability of Syntactic Variants Using Web Snippets. *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP)*, (pp. 537-544). Hyderabad, India.

Harabagiu, S., & Bejan, C. A. (2005). Question Answering Based on Temporal Inference. *Proceedings of the Workshop on Inference for Textual Question Answering.* Pittsburg, Pennsylvania, USA.

Harabagiu, S., & Hickl, A. (2006). Methods for Using Textual Entailment in Open-Domain Question Answering. *44th Annual Meeting of the Association for Computational Linguistics.* Sidney, Australia.

Hartrumpf, S. (2008). Semantic decomposition for question answering. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI)*, pp. 313-317. Patras, Greece.

Hickl, A., Wang, P., Lehmann, J., & Harabagiu, S. (2006). FERRET: Interactive Question-Answering for Real-World Environments. *COLING/ACL Interactive Presentation Sessions.* Sydney, Australia.

Katz, B. (1997). Annotating the World Wide Web Using Natural Language. *Proceedings of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet (RIAO '97).*

Katz, B., & Lin, J. (2003). Selectively Using Relations to Improve Precision in Question Answering. *Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering.*

Katz, B., Borchardt, G., & Felshin, S. (2005). Syntactic and Semantic decomposition Strategies for Question Answering from Multiple Resources. *Proceedings of the AAAI 2005 Workshop on Inference for Textual Question Answering.* Pittsburgh, Pennsylvania, USA.

Laurent, D., Séguéla, P., & Nègre, S. (2005). Cross Lingual Question Answering using QRISTAL for CLEF 2005. *Working Notes for the CLEF 2005 Workshop.* Vienna, Austria.

Laurent, D., Séguéla, P., & Nègre, S. (2006). Cross Lingual Question Answering using QRISTAL for CLEF 2006. *Working Notes for the CLEF 2006 Workshop.* Alicante, Spain.

Laurent, D., Séguéla, P., & Nègre, S. (2008). Cross Lingual Question Answering using QRISTAL for CLEF 2008. *Working Notes for the CLEF 2006 Workshop.* Alicante, Spain.

Lee, Y.-H., Lee, C.-W., Sung, C.-L., Tzou, M.-T., Wang, C.-C., Liu, S.-H., et al. (2008). Complex Question Answering with ASQA at NTCIR 7 ACLIA. *Proceeding of the 7th NTCIR Workshop Meeting*, (pp. 70-76). Tokyo, Japan.

Ligozat, A.-L. (2007). Apport de l'analyse syntaxique des phrases dans un système de questions-réponses. *Traitement automatique des langues , 46*.

Lin, C.-J., & Liu, R.-R. (2008). An Analysis of Multi-Focus Questions. *SIGIR Workshop on Focused Retrieval.* Singapore.

Lin, D., & Pantel, P. (2001). Discovery of inference rules for question-answering. *Journal of Natural Language Engineering , 7* (4), 343-360.

Mel'čuk, I. (1984). *Dependency syntax: theory and practice*. State University of New York Press, Albany.

Moriceau, V., Tannier, X., Grappy, A., & Grau, B. (2008). Justification of Answers by Verification of Dependency Relations - The French AVE Task. *Working Notes for the CLEF 2008 Workshop.* Aarhus, Denmark.

Quintard, L. (2008). *Overview of the Quaero 2008 Monolingual Question Answering Track.* Retrieved from Laboratoire National de Métrologie et d'Essais: http://www.lne.eu/publications_en/research/quaero-QA-2008-overview.pdf

Quintard, L. (2009). *P2 Evaluation Report: Evaluation Design for Task 3.5 on Question Answering Systems.* Quaero program, CTC project.

Rodrigo, A., Peñas, A., & Verdejo, F. (2008). Overview of the Answer Validation Exercise 2008. *Working Notes for the CLEF 2008 Workshop.* Aarhus, Denmark.

Saquete, E., Martinez-Barco, P., R., M., & Vicedo, J. L. (2004). Splitting complex temporal questions for question answering systems. *42nd Annual Meeting of the Association for Computational Linguistics* (pp. 566-573). Barcelone, Spain: Association for Computational Linguistics, Morristown, NJ, USA.

Sun, R., Jiang, J., Tan, Y. F., Cui, H., Chua, T.-S., & Kan, M.-Y. (2005). Using Syntactic and Semantic Relation Analysis in Question Answering. *The Fourteenth Text REtrieval Conference (TREC).* Gaithersburg, MD, USA: Department of Commerce, National Institute of Standards and Technology.

Vallin, A., Giampiccolo, D., Aunimo, L., Ayache, C., Osenova, P., Peñas, A., et al. (2005). Overview of the CLEF 2005 Multilingual Question Answering Track. *Working Notes for the CLEF 2005 Workshop.* Vienna, Austria.

Verberne, S., Boves, L., Oostdijk, N., & Coppen, P.-A. (2007). Discourse-based answering of why-questions. *Traitement Automatique des Langues, Discours et document: traitements automatiques , 47,* 21-41.

Verberne, S., Raaijmakers, S., Theijssen, D., & Boves, L. (2009). Learning to Rank Answers to Why-Questions. *Proceedings of 9th Dutch-Belgian Information Retrieval Workshop (DIR 2009)*, (pp. 34-41).

Yu, H., & Hatzivassiloglou, V. (2003). Towards Answering Opinion Questions: Separating Facts from Opinions and Identifying the Polarity of Opinion Sentences. *Proceedings of 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP),* (pp. 129-136).

http://trec.nist.gov/

http://www.clef-campaign.org/

http://research.nii.ac.jp/ntcir/

http://www.inex.otago.ac.nz/

http://www.quaero.org/

http://pascallin.ecs.soton.ac.uk/Challenges/RTE/

http://nlp.uned.es/clef-qa/ave/