Off-line answer extraction for Question Answering

Mur, Jori

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2008

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.
Chapter 2

Off-line Answer Extraction: Initial experiment

In order to fully understand the task of answer extraction we describe a simple experiment: a state-of-the-art question-answering system is enhanced with an off-line answer extraction module. Simple lexico-POS patterns are defined to extract answers. These answers are stored in a table. Questions are automatically answered by a table look-up mechanism and in the end they are manually evaluated. Not only will this experiment reveal the problems that we want to address in this thesis, it also offers us the possibility of introducing to the reader the framework in which most experiments in the following chapters are performed.

2.1 Experimental Setting

In this section we introduce the question-answering system, the corpus, the question set, and evaluation methods used in our experiments throughout the thesis.

2.1.1 Joost

For all the experiments described in this thesis we use the Dutch QA system Joost. The architecture of this system is roughly depicted in figure 2.1. Apart from the three classical components question analysis, passage retrieval and answer extraction, the system also contains a component called Qatar, which is based on the technique of extracting answers off-line. This thesis focuses on the Qatar component, highlighted by diagonal lines. All components in the system rely heavily on syntactic analysis, which is provided by Alpino, a wide-coverage dependency parser for Dutch. Alpino is used to parse questions as well as the full document collection from which answers need to be extracted. A more detailed description of Alpino follows in section 2.1.2. We will now give an overview of the components of the QA system.

The first processing stage is question analysis. The input for this component is a natural language question in Dutch, that is parsed by Alpino. The goal of question
Chapter 2. Off-line Answer Extraction: Initial experiment

Figure 2.1: Main components and main process flows in Joost.

Analysis is to determine the question type and to identify keywords in the question. Depending on the question type the next stage is either passage retrieval (following the left arrow in the figure) or table look-up (using Qatar following the right arrow).

If the question type matches one of the categories for which we created tables, it will be answered by Qatar. Tables are created off-line for facts that frequently occur in fixed patterns. We store these facts as potential answers together with the IDs of the paragraphs in which they were found. During the question-answering process the question type determines which table is selected (if any).

For all questions that cannot be answered by Qatar, the other path through the QA-system is followed to the passage retrieval component. Previous experiments have shown that a segmentation of the corpus into paragraphs is most efficient for information retrieval (IR) performance in QA (Tiedemann, 2007). Hence, IR passes relevant paragraphs to subsequent modules for extracting the actual answers from these text passages.
2.1. Experimental Setting

The final processing stage in our QA-system is answer extraction and selection. The input to this component is a set of paragraph IDs, either provided by Qatar or by the IR system. We then retrieve all sentences from the text included in these paragraphs. For questions that are answered by means of table look-up, the tables provide an exact answer string. In this case the context is used only for ranking the answers. For other questions, answer strings have to be extracted from the paragraphs returned by IR. Finally, the answer ranked first is returned to the user.

2.1.2 ALPINO

The Alpino system is a linguistically motivated, wide-coverage, grammar and parser for Dutch. Alpino is used to parse questions as well as the full document collection from which answers need to be extracted.

The constraint-based grammar follows the tradition of HPSG (Pollard and Sag, 1994). It currently consists of over 600 grammar rules (defined using inheritance) and a large and detailed lexicon (over 100,000 lexemes) and various rules to recognise special constructs such as named entities, temporal expressions, etc. To optimise coverage, heuristics have been implemented to deal with unknown words and ungrammatical or out-of-coverage sentences (which may nevertheless contain fragments that are analysable). The grammar provides a “deep” level of syntactic analysis, in which wh-movement, raising and control, and the Dutch verb cluster (which may give rise to ‘crossing dependencies’) are given a principled treatment. The Alpino system includes a POS-tagger which greatly reduces lexical ambiguity, without an observable decrease in parsing accuracy (Prins, 2005). The output of the system is a dependency graph, compatible with the annotation guidelines of the Corpus of Spoken Dutch.

A left-corner chart parser is used to create the parse forest for a given input string. In order to select the best parse from the compact parse forest, a best-first search algorithm is applied. The algorithm consults a Maximum Entropy disambiguation model to judge the quality of (partial) parses. Since the disambiguation model includes inherently non-local features, efficient dynamic programming solutions are not directly applicable. Instead, a best-first beam-search algorithm is employed (Van Noord, 2007). Van Noord shows that the accuracy of the system, when evaluated on a test-set of 2256 newspaper sentences, is over 90%, which is competitive with respect to state-of-the-art systems for English.

For the QA task, the disambiguation model was retrained on a corpus which con-
tained the (manually corrected) dependency trees of 650 quiz questions. The retrained model achieves an accuracy on 92.7% on the CLEF 2003 questions and of 88.3% on CLEF 2004 questions.

A second extension of the system for QA, was the inclusion of a Named Entity Classifier. The Alpino system already includes heuristics for recognising proper names. Thus, the classifier needs to classify strings which have been assigned a NAME part-of-speech tag by grammatical analysis, as being of the subtype PER, ORG, GEO or Misc. To this end, lists of person names (120K), geographical names (12K), organisation names (26k), as well as miscellaneous items (2K) were collected. The data are primarily extracted from the Twente News Corpus, a collection of over 300 million words of newspaper text, which comes with annotation for the names of people, organisations, and locations involved in a particular news story. For unknown names, a maximum entropy classifier was trained, using the Dutch part of the shared task for CONLL 2003. The accuracy on unseen CONLL data of the resulting classifier (which combines dictionary look-up and a maximum entropy classifier) is 88.2%.

We have used the Alpino-system to parse the full text collection for the Dutch CLEF QA-task. First, the text collection was tokenised into 78 million words and segmented into 4.1 million sentences. Parsing this amount of text takes over 500 CPU days. Van Noord used a Beowulf Linux cluster of 128 Pentium 4 processors to complete the process in about three weeks. The dependency trees are stored as (25 GB of) XML.

Several components of our QA system make use of dependency relations. All of these components need to check whether a given sentence satisfies a certain syntactic pattern. We have developed a separate module for dependency pattern matching. We will now explain how this matching works.

The dependency analysis of a sentence gives rise to a set of dependency relations of the form \( \langle \text{Head/HPos, Rel, Dep/DPos} \rangle \), where Head is the root form of the head of the relation, and Dep is the head of the constituent that is the dependent. HPos and DPos are string indices, and Rel is the name of the dependency relation. For instance, the dependency analysis of sentence (1-a) is (1-b).

(1)   a. Mengistu kreeg asiel in Zimbabwe

\[1 \text{From the } \text{Winkler Prins spel, a quiz game. The material was made available to us by the publisher, Het Spectrum, bv.} \]
\[2 \text{Various other entities which sometimes are dealt with by NEC, such as dates and measure phrases, can be identified using the information present in POS tags and dependency labels.} \]
\[3 \text{http://cnts.ua.ac.be/conll2003/ner/ dd. April 21, 2008.} \]
\[4 \text{Part of the High-Performance Computing centre of the University of Groningen} \]
2.1. Experimental Setting

English: Mengistu was given asylum in Zimbabwe

\[ Q = \left\{ \begin{array}{l}
\langle \text{krijg/2, subj, mengistu/1} \rangle,
\langle \text{krijg/2, obj1, asiel/3} \rangle,
\langle \text{krijg/2, mod, in/4} \rangle,
\langle \text{in/4, obj1, zimbabwe/5} \rangle
\end{array} \right\} \]

So parsing sentence (1-a) yields 4 dependency relations, three of which contain as head the root of the verb \text{kreeg}. This term appears in the second position of the sentence, and it stands e.g. in a subject relation to the term \text{mengistu} occurring in the first position of the sentence.

A dependency pattern is a set of partially underspecified dependency relations. Variables in a pattern can match with an arbitrary term. In the following example they are represented starting with a capital letter:

\[ \left\{ \begin{array}{l}
\langle \text{krijg/K, obj1, asiel/A} \rangle,
\langle \text{krijg/K, subj, Su/S} \rangle
\end{array} \right\} \]

This pattern matches with the set in (1-b) and would for example instantiate the variable \text{Su} as \text{mengistu}.

2.1.3 Corpus

For all experiments throughout this thesis we use the text collection made available at the CLEF competitions on Dutch QA. From this corpus we will extract the answers. The CLEF text collection contains two years (1994 and 1995) of newspaper text from two newspapers (\text{Algemeen Dagblad} and \text{NRC}) with about 4.1 million sentences in about 190,000 documents. Topics covered range from politics, sports, and science to economical, cultural and social news.

2.1.4 Question Set

The questions we use for our experiments are drawn from a big question pool consisting of questions made available throughout the years by the CLEF organisers. See table 2.1 where we show the different question sets in this pool. The CLEF QA track offers tasks to test question-answering systems developed for languages other than English, including Dutch. Each year, since 2003, it provides new question sets. Moreover, they encourage the development of cross-language question-answering systems. Therefore Dutch source language queries are created to be answered using a target document
<table>
<thead>
<tr>
<th>Question set</th>
<th># Questions</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISEQuA</td>
<td>450</td>
<td>Monolingual Dutch</td>
</tr>
<tr>
<td>CLEF2003NLNL</td>
<td>200</td>
<td>Monolingual Dutch</td>
</tr>
<tr>
<td>CLEF2003NLEN</td>
<td>200</td>
<td>Cross-lingual Dutch English</td>
</tr>
<tr>
<td>CLEF2004NLNL</td>
<td>200</td>
<td>Monolingual Dutch</td>
</tr>
<tr>
<td>CLEF2004NLEN</td>
<td>200</td>
<td>Cross-lingual Dutch English</td>
</tr>
<tr>
<td>CLEF2004NLDE</td>
<td>200</td>
<td>Cross-lingual Dutch German</td>
</tr>
<tr>
<td>CLEF2004NLPT</td>
<td>200</td>
<td>Cross-lingual Dutch Portuguese</td>
</tr>
<tr>
<td>CLEF2004NLES</td>
<td>200</td>
<td>Cross-lingual Dutch Spanish</td>
</tr>
<tr>
<td>CLEF2004NLIT</td>
<td>200</td>
<td>Cross-lingual Dutch Italian</td>
</tr>
<tr>
<td>CLEF2004NLFR</td>
<td>200</td>
<td>Cross-lingual Dutch French</td>
</tr>
<tr>
<td>CLEF2005NLNL</td>
<td>200</td>
<td>Monolingual Dutch</td>
</tr>
<tr>
<td>CLEF2006NLNL</td>
<td>200</td>
<td>Monolingual Dutch</td>
</tr>
</tbody>
</table>

Table 2.1: CLEF question sets

collection in another language as information source. These question sets can also be used to train and test monolingual QA systems for Dutch.

DISEQuA is a corpus of 450 questions translated into four languages (Dutch, Italian, Spanish and English) (Magnini et al., 2003). Setting up common guidelines that would help to formulate good questions, the organisers agreed that questions should be fact-based, asking for an entity (i.e. a person, a location, a date, a measure or a concrete object), avoiding subjective opinions or explanations, and they should address events that occurred in the years 1994 or 1995. It was decided to avoid definitional questions of the form ‘Who/What is X?’ and list questions. Finally, yes/no questions should be left out, too.

CLEF2003NLNL and CLEF2003NLEN are the question sets from the CLEF competitions in 2003 created for the Dutch monolingual task and the Dutch-English cross-language task, respectively. CLEF2003NLNL contained 200 questions of which 180 were drawn from the DISEQuA corpus. The remaining 20 questions were so called NIL queries, i.e. questions that do not have any known answer in the corpus. The CLEF organisers claim that the questions were created independently from the document collection, in this way avoiding any influence in the contents and in the formulation of the questions. For the 2003 cross-language tasks 200 English queries were formulated and it was verified manually that 185 of them had at least an answer in the English text collection. Then the questions were translated into five other languages: Dutch, French, German, Italian and Spanish.
2.1. Experimental Setting

The CLEF2004 question sets for the monolingual and the different cross-lingual tasks each contain 200 Dutch questions (Magnini et al., 2004). Around twenty of the 200 questions in each question set did not have any answer in the document collection. List questions, embedded questions, yes/no questions and why-questions were not considered in this track.

On the other hand, the test sets included two question types that were avoided in 2003: how-questions and definition questions. How-questions may have several different responses that provide different kinds of information, see for example (2).

(2) How did Hitler die?
   a. He committed suicide
   b. In mysterious circumstances
   c. Hit by a bullet

Definition questions are considered very difficult, because although their target is clear, they are posed in isolation, and different questioners might expect different answers depending on previous assumptions. Moreover, it is difficult to determine whether a candidate answer phrase really defines the question term or whether it merely provides some irrelevant information. Definition questions are not regarded as answered by a concise explanation of the meaning of the question term, but rather by providing identifying information that is regarded as interesting. For a definition question asking for a person such as in (3) that means that the answer informs the user for what this person was known. In this case: Elvis was not known for his origins (3-a) although literally this information defines him in the sense that it can only be him to whom the answer refers. Nevertheless, a more informative answer is (3-b). The CLEF approach has tried to keep it simple: Only definition questions that referred to either a person or an organisation were included, in order to avoid more abstract “concept definition” questions, such as ‘What is religion?’.

(3) Who was Elvis Presley?
   a. Only son of Vernon Presley and Gladys Love Smith who was born on January 8, 1935 in East Tupelo.
   b. American pop singer and actor, pioneer of rock & roll music.

From CLEF2005 we only included the question set created for the Dutch monolingual task. The number of questions was again 200 (Vallin et al., 2005). How-questions were not included anymore, since they were considered particularly problematic in the
Chapter 2. Off-line Answer Extraction: Initial experiment

evaluation phase. However, a new subtype of factoid questions was introduced called **temporally restricted questions**. These questions were constrained by either an event (4-a), a date (4-b) or a period of time (4-c). In total 26 temporally restricted questions were included in the question set. Again around 10% of the queries were NIL questions.

(4)  
a. Who was Uganda’s president during Rwanda’s war?  
b. Which Formula 1 team won the Hungarian Grand Prix in 2004?  
c. Who was the president of the European Commission from 1985 to 1995?

From CLEF2006 we also included only the 200 questions created for the Dutch monolingual task (Magnini et al., 2006). For this year list questions were added to the question sets, including both closed list questions asking for a specific finite number of answers (5-a) and open list questions, where as many correct answers could be returned (5-b) as available. The questions were created by annotators who were explicitly instructed to think of “harder” questions, that is, involving paraphrases and some limited general knowledge reasoning, making the task for this year more difficult overall. The restraint on definition questions was lifted, so concept definition questions were now also included.

(5)  
a. What were the names of the seven provinces that formed the Republic of the Seven United Provinces in the 16th century?  
b. Name books by Jules Verne.

Many questions appear in more than one question set. All questions of CLEF2003 are taken from DISEQuA, but there is also overlap between the other sets. In total there are 1486 unique questions.

2.1.5 **Answer set**

The collection of answers was manually created. If the system returned an answer to a question which we judged correct, we added it to the answer set. For those questions for which no correct answer was found, we tried to find the answer manually in the text collection. If still no answer was found, we considered the question a NIL question. That means that we assumed that the text collection did not contain an answer to this question.

We followed the CLEF evaluation guidelines\(^5\) to decide if an answer was correct:

the answer had to be true, supported by the text and exact. One rule was added for which we did not find information in the CLEF guidelines: the answer is considered to be true, if the fact was true in 1994 or 1995. Some facts can change over time. For example, the correct answer to the question ‘Who is the president of France?’ is François Mitterrand until the 16th of May 1995, after that the correct answer is Jacques Chirac. That is why we considered both answers to be correct.

2.2 Initial experiment

In this section we perform a small experiment in which we try to answer questions using off-line answering techniques. Discussing the results we illustrate the main problems with this technique.

2.2.1 Patterns

Before creating patterns for off-line answer extraction we need to determine for which kind of questions we would like to extract answers. Different facets play a role here.

First of all, if we want to cover many questions we had better extract answers to frequently asked questions. During a presentation titled ‘What’s in store for question-answering?’ given in 2000 at the SIGDAT Conference John B. Lowe, then vice president of AskJeeves, shows that Zipf’s law applies to user queries, meaning that a few question types account for a large portion of all question instances (Lowe, 2000). In a tutorial on question-answering techniques for the world wide web Katz and Lin (2003) demonstrate this for questions in the question-answering track of TREC-9 and TREC-2001 as well: ten question types alone account for roughly 20% of the questions from TREC-9 and approximately 35% of the questions from TREC-2001.

Letting the QA-system Joost classify approximately 1000 CLEF questions results also in a Zipf’s distribution, see figure 2.2. The top 3 frequent question types in this figure are, in order of frequency, which-questions (150), location questions (143) and definition questions (124).

This observation leads us to the next three important facets of creating patterns for off-line answer extraction. First, the categorisation of questions depends on the question classification of a particular QA system. Second, answers to a particular class of questions should be suitable for extraction using more or less fixed patterns. Third, the question terms and answer terms of a particular class should be clearly defined.

The most frequent question type in the CLEF question set according to Joost is
the class of *which*-questions. *Which*-questions are questions that start with the term welke or welk (‘which’). Whereas for most question types the type of answer can be derived from the question word (*who*-questions ask for a person, *where*-questions ask for a location), this is not the case for which-questions. Typically the term following ‘which’ indicates what is asked for:

(6)  

a. Which fruit contains vitamin C?  
b. Which ferry sank southeast of the island Utö?

Question (6-a) asks for a type of fruit and question (6-b) asks for the name of a ferry. Lexical knowledge is needed to determine the type of the answer. We can extract this lexical knowledge from the corpus off-line (Van der Plas et al., to appear), but the answers to these types of questions can occur in very different kind of structures. No general pattern can be defined to capture answers to which-questions.

This last issue also holds for location questions. Example questions of this class are given in (7) and their respective answers are given in (8).

(7)  

Location questions  
a. Where is Basra?  
b. Where did the meeting of the countries of the G7 take place?  
c. Where did the first atomic bomb explode?
d. From where did the ferry “The Estonia” leave?

e. Where was Hitler born?

(8) Location answers

a. The city of Basra is only four kilometres away from the border of Iran.
b. (...) the G-7 meeting in Naples.
c. (...) the United States dropped the first atomic bomb on Hiroshima.
d. Two safety inspectors (...) walked around the ferry “the Estonia” for five hours last Sunday before the ship left the harbour of Tallinn.
e. Adolf Hitler was born in the small border town Braunau am Inn.

The answer type of these five questions is clear, they ask for a location. However, a general pattern to extract answers to these questions is hard to discern.

For definition questions of which examples are given in (9) other problems arise. It is possible to come up with a pattern which will extract answers to these kind of questions, the most intuitive being ‘Q is A’. However, extracting facts from newspapers with this pattern also yields very much noise. You will, for example, find sentences such as (10-a) and (10-b). It is very hard to determine automatically whether a candidate answer really defines the term in the question or not.

Another difficulty is to decide how much should be included in the answer. For question (9-c) you would perhaps want to include the whole subordinate clause from sentence (10-c), while for question (9-d) you probably do not want to use the entire appositive in sentence (10-d).

Much work is done especially focusing on this kind of questions (Voorhees, 2003; Hildebrandt et al., 2004; Lin and Demner-Fushman, 2005; Fahmi and Bouma, 2006). Fahmi and Bouma (2006) did use off-line techniques for answering definition questions. They suggest to use machine learning methods after extracting candidate definition sentences to distinguish definition sentences from non-definition sentences. In CLEF and TREC definition questions are considered a special class of questions.

(9) a. Who is Clinton?
b. What is Mascarpone?
c. Who is Iqbal Masih?
d. Who is Maradona?

(10) a. Clinton is the enemy.
b. Mascarpone is the most important ingredient.
c. Iqbal Masih, a 12-year old boy from Pakistan who was known interna-
tionally for his actions against child labour in his country, (...)

d. Diego Maradona, the Argentinian football player who fled the country last week after he shot a journalist, (...)

In addition, we want to point out the following observation. The great advantage of off-line answer extraction and the main reason for being successful in finding the correct answer to a question relies in the fact that it can use frequency counts very easily. However, to be able to use frequency counts both question terms and answer terms should be clearly defined. This advantage disappears with answers such as (10-c), since they probably occur only once in exactly these words.

A category that does meet all four criteria mentioned here (frequent question type, recognised by question classification of the QA system, answers follow fixed patterns, question terms and answer terms are clearly defined) is the class of function questions. It was the category ranked fourth in figure 2.2 with 99 questions. Answers to these kind of questions tend to follow more or less fixed patterns as we will see later in this chapter and the type is recognised by the QA-system we are using. Examples of these questions are:

(11)

a. What is the name of the president of Burundi?
b. Of which organisation is Pierre-Paul Schweitzer the manager?
c. Who was Lisa Marie Presley’s father?

In our first experiment we build a module to find answers to function questions off-line. In following experiments in this thesis we will add more question categories that will at least meet the last three criteria.

The knowledge base we have created to answer function questions is a table with several fields containing a person name, an information bit about the person (e.g., occupation, position) which we call a function, the country or organisation for which the person fullfills this function, and the source document identification. The table lookup mechanism finds the entries whose relevant fields best match the keywords from the question taking into account other factors such as frequency counts.

To extract information about functions, we used a set of surface patterns listed here below. We have split them into two sets to avoid redundancy and improve readability. The first set matches names and functions (e.g. minister Zalm, president Sarkozy). The second set matches functions and organisation or countries (e.g. minister of finance, president of France). Pattern 1 and 3 can be combined with pattern a, c, or d. Pattern 2, 4, and 5 can be combined with pattern b, c, or d.
2.2. Initial experiment

**Person-Function patterns**

1. \(/[\text{FUNCTION}] \ [\text{NAME}_p]/
   
   minister Zalm ‘minister Zalm’

2. \(/[\text{NAME}_p] \ ([\text{TERM}]−) + [\text{FUNCTION}]/
   
   Zalm minister ‘Zalm minister’

3. \(/[\text{FUNCTION}], \ [\text{NAME}_p]/
   
   minister, Zalm ‘minister, Zalm’

4. \(/[\text{NAME}_p], \ ([\text{TERM}]−) + [\text{FUNCTION}]/
   
   Zalm, minister ‘Zalm, minister’

5. \(/[\text{NAME}_p] \ [\text{BE}] \ [\text{DET}] \ ([\text{TERM}]−) + [\text{FUNCTION}]/
   
   Zalm is de minister ‘Zalm is the minister’

**Function-Organisation patterns**

a. \(/[\text{ADJ}] \ [\text{FUNCTION}] /
   
   Franse president ‘French president’

b. \(/[\text{FUNCTION}] \ \text{van} \ [\text{TERM}] /
   
   president van Frankrijk ‘president of France’

c. \(/[\text{FUNCTION}] \ \text{TERM} + \ \text{TERM} + ([\text{TERM}] /
   
   minister Zalm (financiën) ‘minister Zalm (finance)’

d. \(/[\text{TERM}]− \ [\text{FUNCTION}] /
   
   VVD-minister ‘VVD-minister’

[NAME], [ADJ], [VERB], and [DET] match with words in the text that were tagged by Alpino with the respective POS-tags. [TERM] matches with an arbitrary term, [BE] with a form of the verb to be. The subscript P means that the name should be a person name. [FUNCTION] means that we are only interested in function nouns, that is, functions that match with terms such as ‘president’, ‘minister’, ‘soccer-player’ etc. To this end we made a list of function terms by extracting all terms under the node leider ‘leader’ from the Dutch part of Eurowordnet (Vossen, 1998). To extend this list distributionally similar words were extracted semi-automatically. For more details on this technique we refer the reader to Van der Plas and Bouma (2005).
Chapter 2. Off-line Answer Extraction: Initial experiment

We scan the whole text collection for strings matching a pattern. Every time a pattern matches words that match with the boldface terms in the pattern are extracted and stored in a table together with the document ID of the source document.

2.2.2 Questions and answers

We took 99 questions from CLEF2003 and CLEF2004 which were classified by Joost as function questions. Function questions cover two types of questions: person identification questions (12-a) and organisation/country identification questions (12-b).

(12) a. Q. Who was the author of Die Götterdämmerung?
   A. Richard Wagner; Wagner.

b. Of which company is Christian Blanc the president?
   A. Air France.

The complete set of questions and answers can be found in http://www.let.rug.nl/~mur/questionandanswers/chapter2/.

2.2.3 Evaluation methods and results

We evaluate both the extraction task as well as the question-answering task.

The patterns described above were used to extract function facts from the Dutch newspaper text collection provided by CLEF. The output of this process was approximately 71,000 function facts. Approximately 44,000 of these are unique facts. A sample of 100 function facts was randomly selected from the 71,000 extracted facts and classified by hand into three categories based upon their information content similar to Fleischman et al. (2003). Function facts that unequivocally identify an instance’s celebrity (e.g., president of America - Clinton) are marked correct (C). Function facts that provide some, but insufficient, evidence to identify the instance’s celebrity (e.g., late king - Boudewijn) are marked partially correct (P). Function facts that provide incorrect or no information to identify the instance’s celebrity (e.g., oldest son - Pierre) are marked incorrect (I). Table 2.2 shows example function facts and judgements. In table 2.3 we present the results for the extraction task. 58% of the extracted pairs were correct, 27% incorrect, and 15% of the pairs contained some but insufficient information to identify the instance’s celebrity. That means that with confidence level 95% the true percentage of correct pairs lies in the interval (47.8;67.7).

We use these function facts to answer 99 function questions. The answers to these questions are then also classified by hand into three categories: NO ANSWER FOUND
Table 2.2: Example judgements of function facts

<table>
<thead>
<tr>
<th>function fact</th>
<th>Judgement</th>
</tr>
</thead>
<tbody>
<tr>
<td>de voorzitter - Ignatz Bubis</td>
<td>I</td>
</tr>
<tr>
<td>PvdA leider - Kok</td>
<td>C</td>
</tr>
<tr>
<td>Libanese premier - Elias Hrawi</td>
<td>C</td>
</tr>
<tr>
<td>ex-president - Jimmy Carter</td>
<td>P</td>
</tr>
<tr>
<td>NOvAA voorzitter - drs P.L. Legerstee</td>
<td>C</td>
</tr>
</tbody>
</table>

Table 2.3: Evaluation of randomly selected sample of 100 function facts

if no answer was found, **correct** if the selected fact answered the question correctly, and **incorrect** if the selected fact did not answer the question. Results are presented in table 2.4.

<table>
<thead>
<tr>
<th>Evaluation category</th>
<th># Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>No answer found</td>
<td>46</td>
</tr>
<tr>
<td>Correct</td>
<td>35</td>
</tr>
<tr>
<td>Incorrect</td>
<td>18</td>
</tr>
</tbody>
</table>

Table 2.4: Evaluation question-answering task

2.2.4 Discussion of results and error analysis

In this section we discuss the results of the extraction task and the results of the question-answering task.

Starting with the extraction task we see that the precision of the table (58%) was rather low. Since the patterns are very simple and no additional filter techniques were applied, this finding is not surprising. Fleischman et al. (2003) did apply filtering techniques, and they achieved a precision score of 93%.
We cannot calculate a recall score, because we do not know the total number of facts the text contains. Nevertheless, comparing our results with the results from Fleischman et al. may give us some clue assuming that both the newspaper corpora they used and the Dutch CLEF corpus contain approximately the same proportion of useful information. They extracted 2,000,000 facts (930,000 unique) from 15 GB text, that is 1 fact per 7.5 KB. We extracted 71,000 facts (44,000 unique) from 540 MB text, that is 1 fact per 7.6 KB. That means that given the respective precision scores our recall score is probably lower than that of Fleischman et al.. However, the experiment described here should be seen as merely a baseline, described to illustrate problems encountered when performing such a task.

Taking into account which pattern was used to extract which fact in the evaluated random sample of hundred function facts we can determine the quality of the patterns. Table 2.5 presents the evaluation outcome per pattern. The number and letter code for each pattern refers to the numbers and letters on page 27.

<table>
<thead>
<tr>
<th>Pattern</th>
<th># Correct</th>
<th># Partially Correct</th>
<th># Incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1,a} French president Sarkozy</td>
<td>28</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>{1,d} minister Zalm (finance)</td>
<td>13</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>{1,c} VVD-minister Zalm</td>
<td>8</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>{3,a} French president, Sarkozy</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>{3,d} VVD-minister, Zalm</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>{4,b} Sarkozy, president of France</td>
<td>4</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>{4,d} Zalm, VVD-minister</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>{2,b} Sarkozy president of France</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>{2,d} Zalm VVD-minister</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2.5: Result per pattern

The patterns {1,a} and {1,d} account for the largest portion of all correct facts. Pattern {4,b} is the most erroneous pattern and since all other combinations with sub-pattern 4 or sub-pattern b did not result in any correct fact in the sample these sub-patterns had better been left out. The same holds for sub-pattern 2.

Remarkably no fact extracted by a pattern containing sub-pattern 5 appears in the sample. The complete table contains 516 function facts extracted with a pattern containing sub-pattern 5. This sub-pattern is often combined with sub-pattern b. It seems that here the b part is also responsible for many incorrect facts.

Recall that sub-pattern b was $/[\text{FUNCTION}]\ [\text{van}] \ [\text{TERM}]/$. The term after $\text{van}$ in the pattern can be a country or an organisation resulting in a correct fact, but
it turns out that most of the time there appears a determiner in that position. We extracted for example (13-b) from sentence (13-a).

(13) a. Enneus Heerma is de leider van het CDA
    English: Enneus Heerma is the leader of the CDA
b. Enneus Heerma - leider - het

In short, the pattern was too strictly defined to be flexible enough to match correctly examples like (13-a). Manually constructing pattern rules is error prone, and moreover it is a tedious work. A solution would be to learn patterns automatically.

The most striking result for the question-answering task is that the system did not find an answer for almost half of the questions (see table 2.4). In order to find out what caused the fact that answers to these questions were not found we tried to locate the answer to each of these questions manually. We did not do an exhaustive search, but analysed only the first occurrence of the answer we encountered in the corpus. This means that we cannot be sure if the answer might also be found elsewhere. However, the occurrence we encounter first is found randomly, therefore we believe that this error analysis will be representative for the complete set of factors for failing to find an answer. In table 2.6 we list the reasons for not extracting answers.

<table>
<thead>
<tr>
<th>Reason</th>
<th># Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Answer could not be found</td>
<td>14</td>
</tr>
<tr>
<td>Patterns not flexible enough</td>
<td>20</td>
</tr>
<tr>
<td>Reference resolution needed</td>
<td>6</td>
</tr>
<tr>
<td>Knowledge of the world needed</td>
<td>4</td>
</tr>
<tr>
<td>Answer was not a person name</td>
<td>2</td>
</tr>
<tr>
<td><strong>All</strong></td>
<td><strong>46</strong></td>
</tr>
</tbody>
</table>

Table 2.6: Causes for not extracting answers

For 14 questions we could not find an answer at all. In the question set of 2003 and 2004 NIL questions were included, so it is possible that the document collection simply does not contain answers to these 14 questions. For the remaining 32 questions 20 of them were not answered because the patterns were not flexible enough. We illustrate this problem with example (14):

(14) a. Van welk bedrijf is Christian Blanc president?
    English: In which company is Christian Blanc president?
b. Christian Blanc, de president van het zwaar noodlijdende Franse luchtvaartbedrijf Air France, [...]
   English: Christian Blanc, president of the rather destitute French aviation company Air France.

Question (14-a) was one of the 46 questions for which no answer was extracted. Through manual search we found sentence (14-b) in the corpus. Since it is such a long distance from the question terms Christian Blanc and president to the actual answer Air France this sentence did not match with any of the patterns.

For 6 other questions the reason that no answer was found lies in the fact that reference resolution was needed to extract the answer. See for example the sentences in (15-b) containing the answer to question (15-a).

(15) a. Hoe heet de dochter van de Chinese leider Deng Xiaopeng?
   English: What is the name of the daughter of Chinese leader Deng Xiaopeng?

b. Zal (...) Deng Xiaoping (...) het Chinese Nieuwjaar halen? [...] De com-motie in de partijtop zou groot zijn geweest toen Denga jongste dochter, Deng Rong , onlangs zei, dat (...)
   English: Will (...) Deng Xiaoping (...) make it until the Chinese new year? [...] The turmoil among the party leaders was great when Deng’s youngest daughter, Deng Rong, recently said, that (...)

Four questions could only be answered by deductive reasoning, which is still too complicated to do automatically. An example is given in (16).

(16) a. Wie is het staatshoofd van Australië?
   English: Who is the head of state of Australia?

b. In zijn rechtstreeks uitgezonden toespraak zette de Australische premier zijn plan uiteen om een politiek ongebonden president de rol van staatshoofd te laten overnemen van de Britse koningin Elizabeth II.
   English: In his speech broad-casted live the Australian prime-minister explained his plan to transfer the role of the head of state from the British queen Elizabeth II to a politically independent president.

For the two remaining questions it was simply the case that the answer was not a person name. Since we were explicitly looking for a person name, we could never have found the answers to these two questions. One of the two questions is presented here
2.2. Initial experiment

as illustration:

(17) a. Hoe heet de opvolger van de GATT?
   English: What is the name of the successor of the GATT?
   
   b. De WTO begint op 1 januari met haar werk als opvolger van de GATT
      (...) 
   English: The WTO starts working on January 1st as successor of the
   GATT (...) 

For the 53 questions for which we did find an answer 35 (66%) were correct. If we take into account the low precision of the table, which was only 58%, this outcome is better than expected.

For 18 questions the system gave a wrong answer. The largest part of questions answered incorrectly (12 out of 18) were cases such as (18-a) and (18-b):

(18) a. Wie is de Duitse minister van Economische Zaken?
   English: Who is the German minister of Economy?
   
   b. Wie was president van de VS in de Tweede Wereldoorlog?
   English: Who was president of the US during the second World War?

Question (18-a) and (18-b) would be classified as function(minister,duits) and function(president,vs) respectively by question analysis, and thus, in principle can be answered by consulting the function table. Van Economische Zaken and in de Tweede Wereldoorlog are parsed by Alpino as modifiers of the nouns minister and president respectively. These modifiers are crucial for finding the correct answer, but are not recognised by the question analysis, since we defined that the function relation was a binary relation. Thus, question restricted by modifiers are also a serious problem for table extraction techniques, because they do not fit the relation templates.

From the results and error analysis we can conclude that the most important causes for failures are first that the patterns as defined in this experiment are not flexible enough, second that questions restricted by modifiers are hard to answer by this technique. Twenty questions were not answered due to first failure, twelve questions were answered incorrectly due to the second failure. Another problem is that reference resolution is sometimes needed to find the answer. Six questions were not answered due to the lack of reference resolution.

In the following chapters we address these issues. We extend the technique with tables for other question types. However, defining patterns manually takes a lot of
time and effort. Therefore we also included a chapter about how to learn patterns automatically in order to transfer this technique easily to other domains.

2.3 Conclusion

Information extraction is a well-known task in the field of information science. We focus on answer extraction: extracting answers off-line to store them in repositories for quick and easy access during the question-answering process. This technique typically yields high precision scores but low recall scores. We performed a simple experiment: a state-of-the-art question-answering system for Dutch is enhanced with an off-line answer extraction module. Simple lexico-POS patterns are defined to extract answers. These answers are stored in a table. Questions are automatically answered by a table look-up mechanism and in the end they are manually evaluated. This experiment not only revealed the problems that we want to address in this thesis, it also offered us the possibility to introduce the framework in which most experiments in the following chapters are performed.

We introduced the Dutch question-answering system, Joost, which relies heavily on the syntactic output of Alpino. The question set and the corpus we use for our experiments are made available by CLEF. We performed a small experiment in which we tried to answer 99 function questions. For 46 questions no answer was found at all. 35 questions were answered correctly, 18 questions were answered incorrectly. Analysing the results we showed the main problems with the technique of off-line question-answering.

It turned out that the patterns used in the experiment were not flexible enough. A slight difference in the sentence can cause a mismatch with the pattern. In chapter 3 we propose to use dependency-based patterns. Syntactic structures are less influenced by additional modifiers and terms than surface structures.

Modifiers can make a question quite complicated. Temporally restricted questions are well known instances of this class of questions and were introduced in the CLEF question set in 2005. Simple frequency techniques do not work for these type of questions. In chapter 3 we discuss this problem in more detail.

Using reference resolution techniques we would find answers that else would remain undetected. In the experiment described in this chapter we show that some questions would have been answered if this technique was implemented. In chapter 4 we deal with this problem by building a coreference resolution system to increase the number of facts extracted, which leads to more questions being answered.
2.3. Conclusion

Defining patterns by hand is not only a very tedious work, it is also hard to come up with all possible patterns. Automatically learnt patterns will make it easier to adopt this technique for new domains and patterns will be discovered that were not yet anticipated. Chapter 5 will show experiments on this topic.
Chapter 2. Off-line Answer Extraction: Initial experiment