Chapter 5

Extraction based on learned patterns

5.1 Introduction

Defining patterns by hand is not only very tedious work, it is also hard to come up with all possible patterns. By automatically learning patterns we can more easily adapt the technique of off-line answer extraction to new domains. In addition, we might discover patterns that we ourselves had not thought of. Different directions in learning patterns can be taken. Two existing approaches are learning with tree kernel methods and learning based on Bootstrapping algorithms.

Kernel methods compute a kernel function between data instances, where a kernel function can be thought of as a similarity measure. Given a set of labelled instances, kernel methods determine the label of a novel instance by comparing it to the labelled training instances using this kernel function. Tree kernel methods for relation extraction use information based on grammatical dependency trees (Zelenko and Richardella, 2003; Culotta and Sorensen, 2004; Zhao and Grishman, 2005; Bunescu and Mooney, 2005).

Bootstrapping algorithms take as input a few seed instances of a particular relation and iteratively learn patterns to extract more instances. It is also possible to start with a few pre-defined patterns to extract instances with which more patterns can be learned (Brin, 1998; Agichtein and Gravano, 2000; Ravichandran and Hovy, 2002).

We chose to develop a bootstrapping algorithm based on dependency relations. Bootstrapping has the advantage that only a set of seed pairs is needed as input, in contrast to the annotated training data needed for the tree kernel methods. However, we do adopt the use of dependency trees. The algorithm starts with taking a set of tuples of a particular relation as seed pairs. It loops over a procedure which starts by searching for the two terms of a seed pair in one sentence in a parsed corpus. Patterns are learned by taking the shortest dependency path between the two seed terms. The new patterns are then used to find new tuples.

Several bootstrapping-based algorithms to extract semantic relations have been presented in the literature before. Some of them were given as examples above and some of them have been discussed in chapter 1, section 1.2.1 in the light of off-line information extraction. In the following section we will discuss these approaches and more in the context of bootstrapping. The authors describe algorithms that work on different domains and with different levels of automation, but almost all stress the importance of high precision scores by dwelling upon filtering techniques for selecting patterns and instances.

5.1.1 Bootstrapping techniques

Hearst (1992) was the first to describe a bootstrapping method to find patterns and instances of a semantic relation automatically. She focused on the hyponym (IS-A) relation. Starting with three surface patterns including POS-tags discovered by looking through text, she sketches an algorithm to learn more patterns from the instances found by these first three patterns. She did not implement an automatic version of this algorithm, primarily because it was unclear how to find commonalities among the contexts of the newly found occurrences of the seed pairs.

A semi-automatic bootstrapping method was introduced by Brin (1998). Beginning with a small set of five author-title pairs his system DIPRE expanded it to a high quality list of over 15,000 books with only little human intervention (during the iterations bogus books were manually removed from the seed lists). Brin defined simple surface patterns by taking the string starting maximally 10 characters before the first seed term of an author-title pair and ending maximally 10 characters after the second seed term. The number of characters in both cases could be less if the line ends or starts close to the occurrences of the seed terms. To filter out patterns that were too general the pattern's length was used to predict its specificity.

Berland and Charniak (1999) used an algorithm based on the approach of Hearst to extract PART-OF relations. Patterns are found by taking two words in a PART-OF relation and finding sentences in a corpus that have these words within close proximity. The patterns are used to find new instances. The authors introduce a statistical metric for ranking the new instances. Their procedure still included a lot of manual work.

Riloff and Jones (1999) present a multi-level bootstrapping technique, increasing precision by introducing a second level of bootstrapping. The outer bootstrapping mechanism compiles the results for the inner bootstrapping process and identifies the five most reliable terms found by the extraction patterns. These five terms are added

to the collection of previously used seeds. The complete set of terms is used as seed set for the next iteration.

Building on the idea of DIPRE by Brin, Agichtein and Gravano (2000) introduced SNOWBALL, the first system that evaluated both the quality of the patterns and the instances at each iteration of the process without human intervention. They experimented with the HEADQUARTERS RELATION consisting of organisation-location pairs such as Microsoft-Redmond. The patterns include two named entity tags $\langle ORGANISATION \rangle$ and $\langle LOCATION \rangle$, and their context represented as bag-of-words vectors. A pattern's quality is assessed by judging the rate of correctly produced output by comparing it to output of previous iterations. Only the most reliable patterns were kept for the next iteration.

In Ravichandran and Hovy (2002) the authors apply the technique of bootstrapping patterns and instances to answer inter alia birthday questions. They collect a corpus of relevant documents by providing a few manually created seeds consisting of a question term and an answer term to Altavista. Patterns are then automatically extracted from the corpus and standardised. The precision of each pattern is calculated as follows: Take the question term from a seed pair and fit it in at the correct place in the pattern. For example, the pattern ⟨`NAME`⟩ `was born on` ⟨`ANSWER`⟩ becomes `Mozart was born on` ⟨`ANSWER`⟩. Then count the number of times the pattern occurs with the ⟨`ANSWER`⟩ tag matched by any word (C_o) and the number of times the pattern occurs with the ⟨`ANSWER`⟩ tag matched by correct answer term (C_a). The precision of a pattern is then C_a/C_o. For our experiments we will use this evaluation metric. Patterns yielding a high precision score are applied to find the answers to questions during the process of question answering. Remarkably, they do not iterate the process to collect tables of facts. Such structured data would seem very suitable for the process of question answering.

Lita and Carbonell (2004) introduced an unsupervised algorithm that acquires answers off-line while at the same time improving the set of extraction patterns. In their experiments up to 2000 new relations of who-verb types (e.g. who-invented, who-owns, who-founded etc.) were extracted from a text collection of several gigabytes starting with only one seed pattern for each type. However, during the task-based evaluation in a QA-system the authors used the extracted relations only to train the answer extraction step after document retrieval. It remains unclear why the extracted relations are not used directly as an answer pool in the question-answering process. Furthermore, the recall of the patterns discovered during the unsupervised learning stage turned out to be too low.

In Etzioni et al. (2005) the authors present an overview of their information extraction system KnowItAll. Instantiating generic rule templates with predicate labels such as 'actor' and 'city' they create all kinds of extraction patterns. Using these patterns a set of seed instances is extracted with which new patterns can be found and evaluated. KnowItAll introduces the use of a form of point-wise mutual information between the patterns and the extracted terms which is estimated from Web search engine hit counts to evaluate the extracted facts. What is also new in their method is that they use generic template patterns that are instantiated by predicate labels depending on the kind of relation. However, many good extraction patterns do not match a generic pattern. Therefore they included a Pattern Learner to learn domain-specific rules as well. Using the Pattern Learner increased the coverage of their system.

Pantel and Pennacchiotti (2006) describe ESPRESSO, a minimally supervised bootstrapping algorithm that takes as input a few seed instances of a particular relation and iteratively learns surface patterns to extract more instances. They use the Web besides a text corpus of 6.3 million words to increase recall. To evaluate the patterns the authors calculate an association score between a pattern and highly reliable instances based on point-wise mutual information. To evaluate the instances the method works the other way around: calculating an association score between an instance and highly reliable patterns.

In contrast to the bootstrapping approaches discussed above we learn patterns based on dependency relations instead of surface patterns. Using dependency relations we can simply search for the shortest path between the two terms in the dependency tree. Surface patterns are typically harder to define in a natural and meaningful way. For Hearst this was the reason to not implement her algorithm. Brin used a window of maximally ten characters before the first seed term and maximally ten characters after the second seed term, which may result in patterns starting or ending in the middle of a word. Ravichandran and Hovy and Pantel and Pennacchioti apply a complex method based on a suffix tree constructor. The difficulty here is to find suffixes that are frequent but at the same time not too general.

Another drawback of surface patterns, already explained in chapter 3 is that surface patterns are not able to handle long-distance dependencies. It is one of the shortcomings listed by Ravichandran and Hovy in their paper: For the question "Where is London?" their system could not locate the answer in the sentence "London, which has one of the most busiest airports in the world, lies on the banks of the river Thames" due to the explosive danger of unrestricted wild-card matching as would be required in

a matching pattern. Dependency patterns experience no difficulties in detecting such long-distance dependencies.

The results of bootstrapping techniques can be used for different purposes. Semantic taxonomies such as WordNet can automatically be constructed and extended. Ravichandran and Hovy use the learned patterns in their question answering system.

We also intend to use the results for question answering, but in a different way. We will use bootstrapping techniques to collect answers off-line. Starting with ten seed answers we automatically find patterns which can be used to find new answers. The collected answers can be used in the off-line module of a question-answering system as described in previous chapters.

In the discussion of the literature all kinds of relations came up, from general relations such as hyponym relations and part-of relations to more specific relations such as author-title relations, headquarter relations, and birth date relations. For our experiments we are interested in relations suitable for off-line question answering. In principle, this includes all the relations mentioned above, depending on the kind of questions you wish to answer. We will restrict ourselves, however, to about three relations, namely CAPITAL relations, FOOTBALL relations, and MINISTER relations. More details about our experiments can be found in section 5.3.

To learn patterns using bootstrapping techniques it is imperative to make a good start. A good seed list is essential. However, for some relations it is easier to come up with a good set of seeds than for other relations. For instance, for the capital relation it is very easy to create a list of seed-pairs, since the seeds of a pair are typically in a one-to-one relation to each other and each consists of only one term. In contrast to manner-of-death facts which are facts describing how someone died. The manner of death can be formulated in all kind of ways and therefore it is not obvious how to formulate the seed pairs. We discuss this issue further in section 5.5.

5.1.2 Aims and overview

A crucial issue for bootstrapping-based algorithms is to ascertain that the patterns and instances found are reliable in order to avoid the extraction of much noise during further iterations. Except for Etzioni et al. who try to improve KnowItAll's recall and Pantel and Pennacchiotti who try to boost their recall by using general patterns and the Web, all work discussed above on relation extraction is focused more on precision than on recall. Many of the previously mentioned methods describe filtering techniques for both patterns and instances to preserve high precision scores. Brin

states that a recall of just 20% may be acceptable, but an error rate over 10% would likely be useless for many applications. He says that each pattern need only have very small coverage when using the web as resource, since it is so vast that the total coverage of all the patterns can still be substantial.

It seems natural to focus on precision when working with bootstrapping techniques. Many unreliable patterns will result in noisy sets of instances, which in return will yield wrong patterns which will extract incorrect instances and so on. The question we are interested in here is if this view also holds when we apply bootstrapping techniques for *off-line* question answering. Do we still need to focus on precision or might the balance between precision and recall be different? After all, storing the answers offline opens up the possibility of applying filtering techniques afterwards, for instance by using frequency counts. Therefore, there is no need to focus on precision during the extraction process. On the contrary, we have to make sure that in any case the correct answer is among the facts extracted.

Ravichandran and Hovy applied the technique of bootstrapping for QA. In their experiments the patterns are used to find answers during the *online* process of question answering. First a set of relevant documents is retrieved by feeding the question terms into a search engine. The following step is matching the learned patterns to the documents, thus finding candidate answers. The system does not apply a threshold for precision, all patterns found are used to select candidate answers. The precision scores of the patterns are used to rank the candidate answers. Answers found by highly precise patterns are ranked higher.

However, we believe that this technique is also pre-eminently suitable for *off-line* answer extraction. We start with ten hand-crafted facts, which we use to find patterns in a corpus. Applying these patterns again to the corpus we can find new facts. This process is iterated until a certain stopping condition is reached and then the extracted facts can be stored before we start the question answering process. An important additional benefit of the off-line technique is that we do not need to focus on precision during the bootstrapping procedure. The filtering of noise can be done afterwards.

This observation might alter perspectives on recall and precision for relation extraction using bootstrapping techniques. As we have seen, previous studies in this area focus on high precision scores. Typically, a high precision score is achieved by evaluating the patterns and the instances and selecting only those that meet a certain reliability threshold for the next iteration. Although we cannot ignore precision completely if we want to avoid generating too much noise, we probably can get by with less control on precision.

In this chapter we show that aiming at a high precision score during bootstrapping is not always beneficial for question answering. Our hypothesis is that for off-line answer extraction low precision scores will not hurt performance of question answering while it will greatly benefit from high recall scores.

We define a bootstrapping algorithm in which we can employ different levels of precision by changing the threshold for pattern selection. The algorithm is described in section 5.2. The experiments are evaluated on the question answering task. More details are given in section 5.3. In section 5.4 and section 5.5 we discuss our findings and we conclude in section 5.6.

5.2 Bootstrapping algorithm

We present a bootstrapping algorithm for finding dependency-based patterns and extracting answers. The algorithm takes as input ten seed facts of a particular question category. For example, for the category of capital questions we feed it ten country-capital pairs. On the basis of these ten seed pairs patterns are learned. We use these patterns to find new facts and these new facts can again be used to deduce patterns.

The algorithm iterates through three separate stages: the pattern-induction stage, the pattern-filtering stage and the fact-extraction stage. To explore the space in which to find the ideal balance between recall and precision we can change the threshold that is employed to select the new patterns in the pattern-filtering stage. Setting the threshold high means we only select highly reliable patterns for extraction resulting in highly precise tables of facts. Setting the threshold lower allows more patterns to pass the filtering stage, which will result in more facts being extracted, but also more noise. In the next three sections we describe the stages in more detail.

5.2.1 Pattern induction

We start with ten seeds. As an example we take the seed pair *Riga* and *Letland* (Latvia) to find patterns to extract capital-country facts. We search in the corpus for sentences containing both these terms. The first sentence we find is for example as follows:

(1) Riga, de hoofdstad van Letland, is een mooie stad
English: Riga, the capital of Latvia, is a beautiful city.

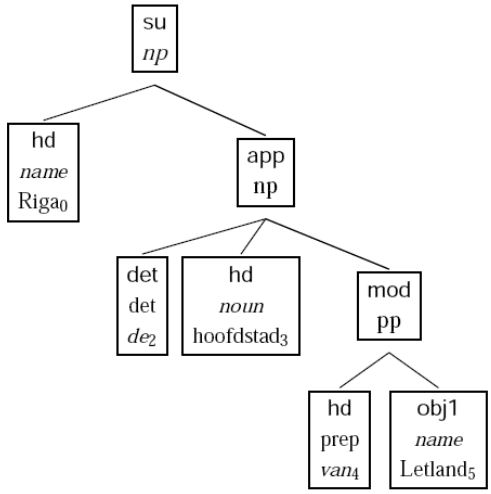

Figure 5.1: Riga, de hoofdstad van Letland

We deduce the pattern from a parsed corpus, so we can start with selecting the shortest dependency path between the term *Riga* and the term *Letland*. The relevant part of the dependency tree for sentence (1) is shown in figure 5.1. Snow et al. (2004) also added optional "satellite links" to each shortest path, i.e. single links not already contained in the dependency path added on either side of each seed term. For general patterns this can be helpful: Snow et al. search for hypernym relations. The shortest path between to nodes of a hypernym relation might not contain enough information. They wished to include important function words like 'such' (in "such NP as NP") or 'other' (in "NP and other NPs") into their patterns. We implemented this option as well. However, for specific relations such as employed for open-domain question answering they are not needed. Almost all patterns we found are without satellites. And if a pattern included satellites its precision typically did not meet the threshold.

Represented in triples the path from *Riga* to *Letland* looks like this:

(2) $\left\{ \begin{array}{l} \langle \texttt{riga/0, app, hoofdstad/3} \rangle, \\ \langle \texttt{hoofdstad/3, mod, van/4} \rangle, \\ \langle \texttt{van/4, obj1, letland/5} \rangle, \end{array} \right\}$

Replacing the seed terms and indices by variables we get the following pattern:

$$(3) \quad \left\{ \begin{array}{l} \langle \texttt{Capital/Ca, app, hoofdstad/H} \rangle, \\ \langle \texttt{hoofdstad/H, mod, van/V} \rangle, \\ \langle \texttt{van/V, obj1, Country/Co} \rangle, \end{array} \right\}$$

We have added part-of-speech constraints in the patterns. This means that for terms to match with the `Capital` or `Country` variable they ought to be names.

In this way we find more dependency patterns with the same seed terms. Similarly we use nine other seed pairs to find patterns. In the end, when the whole corpus is searched through, we have collected a whole set of patterns. This set of patterns is transferred to the pattern-filtering stage.

5.2.2 Pattern filtering

In the pattern-filtering stage we only keep the reliable patterns. To find the optimal balance between precision and recall for off-line answer extraction we perform three different bootstrapping experiments in which we vary the precision threshold for selecting patterns. The precision of a pattern is calculated in the same way as in Ravichandran and Hovy (2002). Instead of replacing both seed terms by a variable as in (3), we now replace only the answer term with a variable. In our Riga example the pattern will then become:

$$(4) \quad \left\{ \begin{array}{l} \langle \texttt{Capital/Ca, app, hoofdstad/H} \rangle, \\ \langle \texttt{hoofdstad/H, mod, van/V} \rangle, \\ \langle \texttt{van/V, obj1, letland/L} \rangle, \end{array} \right\}$$

For each pattern obtained in the pattern-induction stage, we count how many times in the corpus the variable matches with any word and we count how many times the variable matches with the correct answer term according to the seed list. Then the precision of each pattern is calculated by the formula $P = C_a/C_o$, where C_o is the total number of times the pattern matched a phrase in the corpus and C_a is the total number of times the pattern matched a phrase in the corpus containing the correct answer term.

All patterns we select have to be found more than once. For the most lenient version of our algorithm the additional precision threshold is zero, which means that we keep all patterns with a frequency higher than one. If we want to give more weight

to precision we make the filtering process more strict: Not only does a pattern have to occur more than once, its precision score also has to meet a certain threshold, τ_p.

In the second experiment we set threshold τ_p at 0.50 and in the third experiment at 0.75, which means that we select only those patterns that were found more than once and that have a precision score higher or equal to 0.50 or 0.75 respectively. In this way we can alter the pattern filtering procedure from very lenient to very strict. Using very lenient patterns will result in extracting many facts, i.e. higher recall, using very strict patterns will result in extracting mostly correct facts, i.e. higher precision.

The patterns that remain are used in the next stage, the fact-extraction stage.

5.2.3 Fact extraction

The patterns that have passed the filter in the previous stage are matched against the parsed corpus to retrieve new facts. After retrieval we select a random sample of one hundred facts and manually evaluate it. If more than τ_f facts are found the iteration process is stopped, else all facts are used without any filtering as seeds again for the pattern-induction stage and the process repeats itself. In our experiments we have set τ_f to 5000. This number depends on the size of the corpus, the kind of relation, and the amount of time there is to process the learning algorithm.

5.3 Experiment

The corpus we use in our experiments is the same corpus used in experiments described in previous chapters, the CLEF corpus.

We selected three different question types, capital, football, and minister. Capital and football are binary relations, respectively between a location (France) and its capital (Paris) and between a soccer player (Dennis Bergkamp) and his club (Ajax). The minister facts are triple relations between a name (William Perry), a department (Defence) and a nationality (American) or a country (United States).

The reason we chose the capital relation was that it is a very straightforward relation: each country typically has one capital and each capital belongs to one country. The minister and football relations were not yet introduced in earlier chapters. We chose the football category because we believed that the corpus would contain a lot of information about football players. We chose the minister relation since we encountered minister questions in the CLEF question set. In earlier experiments they were classified as function questions, but then a slot was missing in the relation. Func-

tion relations are between a function and a nationality: French president. Minister relations are between a function, a nationality and a department: French minister of foreign affairs. Therefore, we wanted to create a separate table for minister questions.

Since a minister fact comprises three terms, patterns for extracting minister facts differ slightly from the patterns for binary relations. In this case we select two shortest dependency paths: one between the person name and the nationality and one between the person name and the department.

For each of the three question types we created ten seed facts which are listed in table 5.1, table 5.2, and table 5.3. Some seeds we thought of ourselves, some were discovered by looking through the corpus, and some were identified by looking at wikipedia sites.

The initial seeds should be chosen very carefully since they form the basis of the learning process. In the set of capital seeds we have for example also included the capital of a province (Drenthe). In addition, we have used both the adjective form of the country as well as the noun itself to cover a greater variety of patterns. We also had European-Brussels as a seed pair, since this pair occurred very frequently in the corpus. However, during experimentation we found out that Europe has many culture capitals, so it would perhaps have been better to not include this seed pair.

For the football seeds and minister seeds we had to take good care that the seeds represented facts true in 1994 and 1995, since those are the years covered by the CLEF corpus. Complete names and last names were used, since that is how the football players are referred to in the corpus.

For the minister seeds it turned out that due to the way of parsing, the departments consisted of maximally one term. For that reason we have the seed *zaken* 'affairs' instead of *buitenlandse zaken* 'foreign affairs'. This is not ideal, but we think it will work for it will match with the parsing result of the question and using the `d-score` described in chapter 3 which calculates the overlap in dependency relations between the question and the answer sentence we will find the correct kind of affairs. This issue is discussed further in section 5.4, page 126.

Last but not least, as Ravichandran and Hovy (2002) also point out, the seeds have to be selected so that the questions they represent do not have long lists of possible answers, as this would affect the confidence of the precision scores for each pattern.

After we found ten sound seed pairs we retrieved all sentences from the CLEF corpus containing at least one of the seed pairs. It did not matter in which order both seed terms appeared, nor was the search case sensitive. All sentences were parsed by the dependency-parser Alpino. The two seed terms of a seed pair were localised in

Country	Capital
Amerikaanse	Washington
Bulgaarse	Sofia
Drenthe	Assen
Duitsland	Berlijn
Europese	Brussel
Frans	Parijs
Italiaans	Rome
Bosnisch	Sarajevo
Rusland	Moskou
Spaans	Madrid

Table 5.1: Ten Capital seeds

Person	Club
Litmanen	Ajax
Marc Overmars	Ajax
Wim Jonk	Inter
Dennis Irwin	Manchester United
Desailly	AC Milan
Romario	Barcelona
Erwin Koeman	PSV
Jean-Pierre Papin	Bayern München
Roberto Baggio	Juventus
Aron Winter	Lazio Roma

Table 5.2: Ten Football seeds

each sentence. Then we sought the shortest path between these two terms. Optionally satellite links were added, but as explained earlier, these satellite links did not improve the performance of the experiments.

We selected only those patterns with a frequency higher than one and we added part-of-speech constraints depending on the question type. For the football patterns, for instance, both question term (a football player) and answer term (a football club) should be names. In the most lenient experiment all these patterns were used to extract new facts in the CLEF corpus.

For the two other experiments we applied an additional selection step. For one experiment we took from the set of patterns with a frequency higher than one only

Person	Department	Nationality
Warren Christopher	zaak	Amerikaans
Javier Solana	zaak	Spaans
Klaus Kinkel	zaak	Duits
Sorgdrager	justitie	Nederlands
Melkert	zaak	Nederlands
Helseltine	handel	Brits
Chi Haotian	defensie	Chinees
Juppé	zaak	Frans
Willy Claes	zaak	Belgisch
Jorritsma	verkeer	Nederlands

Table 5.3: Ten Minister seeds

those patterns with a precision equal to or higher than 50%. For the other experiment we took from the set of patterns with a frequency higher than one only those patterns with a precision equal to or higher than 75%. We calculate the precision of the patterns as described in section 5.2.2. The patterns with a precision that meets the threshold are used to find new facts in the CLEF corpus.

This process is repeated for two iterations or until we find more than 5000 facts. Then the process is stopped after the fact-extraction stage. We chose these stopping conditions for practical reasons; especially the pattern filtering stage can take very long to process if there are many seeds involved.

The tables were used to answer questions that Joost classified into one of the three categories, capital, football and minister. Since we wanted to do the evaluation on more questions than were available in the CLEF set, we created some ourselves. To find extra capital questions we typed into Google the query *Wat is de hoofdstad van* 'What is the capital of'. Football questions were created by asking five people names of famous football players in 1994 and 1995. With each name we filled a template question: *Bij welke club speelde X?* 'For which club did X play?'. To create extra minister questions we typed into Google the query *Wie is de minister van* 'Who is the minister of'. Nationalities were added at random. In the end we had 42 capital questions, 66 football questions and 32 minister questions. We checked for all if the answer appeared in the corpus. A few example questions with their answers are given in table 5.4. The complete set of questions is listed in `http://www.let.rug.nl/~mur/questionandanswers/chapter5/`.

Question	Answers
Wat is de hoofdstad van Canada?	Ottawa
Wat is de hoofdstad van Cyprus?	Nicosia
Wat is de hoofdstad van Haïti?	Port-au-Prince
Bij welke club speelt Andreas Brehme?	1. FC Kaiserslautern
Bij welke club speelt Aron Winter?	Lazio Roma; Lazio
Bij welke club speelt Baggio?	Juventus; Milan
Wie is de Poolse minister van financiën?	Grzegorz Kolodko; Kolodko
Wie is de Nederlandse minister van visserij?	Bukman; Van Aartsen; van Aartsen
Wie is de Japanse minister van buitenlandse zaken?	Yohei Kono

Table 5.4: Sample of question used for evaluation

5.3.1 Evaluation

We evaluated the tables by implementing them into Joost as Qatar tables. We let Joost run over all questions en we counted how many questions were answered by the module Qatar. The system returned a top five list of answers. We counted how many times the correct answer ended up at rank one and we calculated the mean reciprocal rank, i.e. for each question we took the reciprocal of the rank at which the first correct answer occurred or 0 if none of the returned answers was correct. Then we took the average over all questions. All questions had a correct answer somewhere in the corpus.

5.3.2 Results

The results are shown in table 5.5, 5.6, and 5.7. For each category we have first listed the results for the experiments where we selected all patterns found with a frequency higher than one. This is the most lenient experiment with regard to precision. For the category capital we found 39 patterns in the first round. These are found using the ten seed pairs from table 5.1. Applying these 39 patterns we extracted 3405 new facts. We estimated the precision based on a random sample of hundred facts to be 58%. When using these table as Qatar module in Joost, we answered 35 of 42 questions correctly. The mean reciprocal rank was 0.865. For the second round we repeated the process using 3405 facts. With these facts we found 1306 patterns as presented in the

table. The 1306 patterns in turn returned 234,684 facts.

The middle part of the tables show the results for the experiment in which we filter the patterns using a precision threshold of 50%. The last part of the tables finally lists the results for the experiment in which we apply a filter to select only those patterns that meet a precision threshold of 75%. For the football experiments and the minister experiments we stopped the iteration of the process after we found more than 5000 facts, for the capital experiments we stopped after two iterations.

The best performance per category is marked in bold. For the minister results there was no difference in performance, therefore none of the outcomes is marked in bold.

5.4 Discussion of results

From the data in table 5.5 about extracting capital facts we can see that using no precision level at all for pattern selection results in the extraction of so much noise (the precision of a randomly selected sample was only 1%) that it hurts performance of the QA task. Only 14 out of 42 questions were answered correctly.

However, the data in this table also show that we do not need high precision tables to get the best results, since the best results for answering capital questions was obtained in the second round of iterations with pattern selection at precision level 0.50. The precision of the extracted facts was only mediocre: 49%, compared to 83% in the experiments with pattern selection at precision level 0.75. The number of facts, on the other hand, was almost twice as high (4123 vs. 2344). As it turns out, that was the decisive factor here. 37 out of 42 questions were answered correctly with a mean reciprocal rank of 0.895.

Analysis of the football table (5.6) shows that it can even be more extreme. The best result is again for an experiment at the pattern selection precision level 0.50, but this time there is no significant difference with the result for the experiment with pattern selection at precision level 0. Large numbers of incorrect facts were extracted (the precision of the randomly selected samples also was only 1% here), still it did not hurt performance of the QA task.

This rather contradictory result can be explained by the patterns that were found for the extraction of football facts. A very frequent pattern we found was (5).

(5) $\left\{ \langle \texttt{NameFootballplayer/Nf, mod, NameClub/Nc} \rangle, \right\}$

Capital Freq: > 1	# patterns	# facts (Prec)	# Correct answers	MRR
1st round	39	3405 (58%)	35 of 42	0.865
2nd round	1306	234,684 (1%)	14 of 42	0.510
Prec: $\geq 50\%$; Freq: > 1				
1st round	24	2875 (63%)	35 of 42	0.851
2nd round	171	4123 (49%)	**37 of 42**	**0.895**
Prec: $\geq 75\%$; Freq: > 1				
1st round	17	2076 (83%)	35 of 42	0.839
2nd round	64	2344 (83%)	35 of 42	0.851

Table 5.5: Results for learning capital patterns (Best results in bold)

Football Freq: > 1	# patterns	# facts (Prec)	# Correct answers	MRR
1st round	19	115,296 (1%)	40 of 66	0.659
Prec: $\geq 50\%$; Freq: > 1				
1st round	11	109,435 (1%)	**41 of 66**	**0.667**
Prec: $\geq 75\%$; Freq: > 1				
1st round	6	196 (26%)	11 of 66	0.167
2nd round	28	31,958 (2%)	18 of 66	0.312

Table 5.6: Results for learning football patterns (Best results in bold)

Minister Freq: > 1	# patterns	# facts (Prec)	# Correct answers	MRR
1st round	3	5425 (71%)	23 of 32	0.745
Prec: $\geq 50\%$; Freq: > 1				
1st round	2	4380 (69%)	23 of 32	0.745
2nd round	27	5670 (63%)	23 of 32	0.745
Prec: $\geq 75\%$; Freq: > 1				
1st round	2	4380 (69%)	23 of 32	0.745
2nd round	18	5463 (62%)	23 of 32	0.745

Table 5.7: Results for learning minister patterns

where `NameFootballplayer` and `NameClub` are two variables that should match with names. The pattern matches for example with the phrase in (6).

(6) Jari Litmanen (Ajax).

However, the pattern is so general that it matches with a lot of phrases in the corpus. For example with abbreviations such as (7):

(7) Rijksuniversiteit Groningen (RUG).

It matches with many other kind of terms as well. That is why we extract so many incorrect facts. Yet, these incorrect facts typically have nothing to do with football players, and thus they do not cause incorrect answers to football questions.

No significant differences were found between the results of the experiments performed for the minister category. The reason for this outcome is that only one highly precise pattern (8) was responsible for the extraction of the majority of facts. On its own it extracted 4346 facts already. The precision of this pattern as calculated by the formula $P = C_a/C_o$ described in section 5.2.2 was 0.86. Although other patterns are found, they do not contribute enough to make a difference in the result for the QA task.

$$(8) \quad \left\{ \begin{array}{l} \langle \texttt{minister/M, app, Name/N} \rangle, \\ \langle \texttt{minister/M, mod, van/V, obj1, Department/D} \rangle, \\ \langle \texttt{minister/M, mod, Nationality/N} \rangle, \end{array} \right\}$$

We can conclude that the results of the experiments for the extraction of capital and football facts suggest that for the benefit of off-line QA we better focus on high recall rather than on high precision. This is not contradicted by the results for the minister experiments. However, the balance between precision and recall can differ over different categories, depending on the kind of patterns discovered. In our experiments the patterns for the football category extracted a lot of noise, but it did not hurt the performance. For the capital category we had to be more careful.

Although for each question the corpus contained the correct answer, not all questions were answered. We did an error analysis to investigate which problems underlay these omissions.

We frequently observed mismatches between a question term and an answer term. Examples include *Burundisch* vs. *Burundees*, *Sicilië* vs *Siciliaans*, *Tjetjenië* vs. *Tsjetjenië*. This is not a problem specific for learning patterns of course, it is rather a general question answering problem: mismatching between terms in the question and terms in the answer. We had created a list of location name variants to cover the variety of forms in which a name of a location can appear (for example, starting with a capital or starting with a lower-case letter, as adjective or as nominative, different ways of spelling, etc.). The list contains 204 entities with an average of 5.6 variants. Unfortunately, it appeared that some variants were still missing.

It turned out that the departments for the minister questions still pose a problem. The `d-score` between the question and the sentence which contains the answer is used as an additional factor in conjunction with frequency in determining whether an answer is correct. In the case of a question asking about a minister of social affairs or economic affairs using frequency only to determine the correct answer would give the wrong result, since the most frequent minister of any affairs in newspaper text is typically the minister of foreign affairs. We hoped that a score that combines frequency and `d-score` would return the correct answer. In some cases this indeed helped, but in others the difference in frequency was so large, that in spite of the `d-score` the wrong answer still popped up. In the case of capital questions it once went the other way around: For the question 'What is the capital of France?' the system returned the answer *Belfort*, found in the sentence: *[...] dat Belfort de afgelopen maand de sociale*

hoofdstad van Frankrijk is geweest, '[...] that Belfort had been the social capital of France last month'. In spite of a higher frequency (20 vs. 1) the correct answer was not given due to a high `d-score` for the incorrect answer. The solution for minister questions might be that the departments are treated as one term. That will give an exact match with the question term and only the frequency of the minister with the correct department is taken into account.

A less serious problem for minister questions was that our list of questions[1] contained questions about Dutch ministers while the nationality was missing in the answer sentence, which makes sense using a Dutch newspaper corpus. It is likely that user questions on Dutch corpora would not include the nationality when asking about a Dutch minister.

Most errors for football questions were due to the fact that the system had not derived the pattern needed to extract the answer. More iterations might be needed and here we encounter the drawback of extracting too much noise. Although the noise did not hurt performance in the sense that it selected incorrect answers, it did make it hard to perform a next iteration, since it cost days to process as many seeds as we found. Instead of taking all facts found as seeds for the next iteration we could select a sample of only correct facts. This would not only reduce the amount of processing to be done, it would also help to reduce the amount of incorrect patterns derived.

5.5 General discussion on learning patterns

The difficulty about learning patterns is that you have to come up with a set of good seeds. For the categories we used in our experiments that can be done quite easily. However, for a category such as inhabitants, for example, it becomes much harder, since the answer to an inhabitants question does not always contain one term (e.g. 800,000). Sometimes it consists of two terms (e.g. 8 million) and sometimes it consists of even more terms (e.g. almost 8 million). The goal is then to find patterns that are able to handle this, that extract sometimes one term and in other cases several terms that should be combined to one answer term. To do this automatically during the process of learning is not easy.

Another problem could occur with inhabitant seeds during the calculation of the precision of the patterns. To calculate the precision of the pattern we counted how many times in the corpus the pattern matches with the correct answer term according to the seed list. Since there are so many variations of answers that can be considered

[1] See `http://www.let.rug.nl/mur/questionsandanswers/chapter5/ministerquestions.txt`

correct (e.g. *798,345*, *800,000*, *800 thousand*, *almost 800,000*, etc), judging an answer to be correct only when it precisely matches the seed term will result in discarding correct patterns.

In addition, we saw that a pattern can work very well on the seed list (see as an example pattern (5) above), even if it in general extracts many incorrect facts. For example, for the capital category we found pattern (9). Using this pattern with the `Country` variable matching with *French*, the `Capital` variable will match with *Paris* (e.g. *the French championships in Paris*). Still this is a noisy pattern, because when `Country` matches with *European* (e.g. *the European championships in Berlin*) or with *national* (e.g. *the national championships in Heerenveen*) we retrieve incorrect facts.

(9) $\left\{ \begin{array}{l} \langle \texttt{kampioenschap/K, mod, Country/C} \rangle, \\ \langle \texttt{kampioenschap/K, mod, in/I, obj, Capital/Ca} \rangle, \end{array} \right\}$

This raises doubt as to the value of this evaluation measure.

An alternative measure could be the one introduced by Pantel and Pennacchiotti (2006). They define the reliability of a pattern as its average strength of association across each input instance weighted by the reliability of each instance. The formula is based on point-wise mutual information (PMI) (Cover and Thomas, 1991).

According to Blohm et al. (2007) who compared different filtering functions for evaluating extraction patterns PMI-based filtering yields a high recall. So this might be an appropriate evaluation measure for the filtering of learned patterns for answer extraction. Further investigation needs to be done to determine the precise impact of other evaluation measures on this task.

Next, we give some examples of patterns we learned, which we might not have thought of ourselves, together with some matching phrases. For extracting capital facts we found:

(10) a. $\left\{ \begin{array}{l} \langle \texttt{open/O, su, president/P, mod, Country/C} \rangle, \\ \langle \texttt{open/O, mod, in/V, obj, Capital/Ca} \rangle, \end{array} \right\}$

b. [...] Radio Free Europe, is gisteren in Praag geopend door de Tsjechische president Havel. (AD, September 9th 1995)
English: Radio Free Europe was opened yesterday in Prague by the Czech president Havel.

c. De Internationale Conferentie over Bevolking en Ontwikkeling die vanmorgen in Kaïro door [...] de Egyptische president Mubarak werd geopend

[...] (NRC, September 5th 1994)
English: The International Conference on Population and Development which was opened this morning in Cairo by the Egyptian president Mubarak [...]

(11) a. $\left\{ \begin{array}{l} \langle \texttt{ben/C, predc, stad/S, mod, van, obj, Country/C} \rangle, \\ \langle \texttt{ben/C, mod, na/N, obj, Capital/Ca} \rangle, \end{array} \right\}$

b. Sint-Petersburg is met vijf miljoen inwoners na Moskou de grootste stad van Rusland [...] (NRC, April 20th 1995)
English: Saint Petersburg is with five million inhabitants the largest city of Russia after Moscow [...]

c. De staat Córdoba - de gelijknamige hoofdstad is na Buenos Aires de grootste stad van Argentinië - [...] (NRC, June 26th 1995)
English: The state Córdoba - the capital of the same name is the largest city of Argentina after Buenos Aires - [...]

For extracting football facts we found:

(12) a. $\left\{ \begin{array}{l} \langle \texttt{ben/C, su, Name/N} \rangle, \\ \langle \texttt{ben/C, predc, man/M, mod, bij/B, obj, Club/C} \rangle, \end{array} \right\}$

b. Wim Jonk was de beste man bij Inter. (AD, March 21st 1994)
English: Wim Jonk was the best man of Inter.

c. Romario was de grote man bij Barcelona. (NRC, May 2nd 1994)
English: Romario was the big man of Barcelona.

For extracting minister facts we found:

(13) a. $\left\{ \begin{array}{l} \langle \texttt{minister/M, mod, van/V, obj1, Department/D} \rangle, \\ \langle \texttt{zeg/Z, su, minister/M, app, Name/Na} \rangle, \\ \langle \texttt{zeg/Z, mod, voor/V, obj1, televisie/T} \rangle, \\ \langle \texttt{televisie/T, mod, Nationality/N} \rangle, \end{array} \right\}$

b. Minister van Europese zaken Alain Lamassoure zei gisteren voor de Franse televisie [...] (NRC, January 3rd 1994)
English: Minister of European affairs Alain Lamassoure said yesterday in front of the French television [...]

c. Minister van buitenlandse zaken Warren Christopher zei voor de Amerikaanse televisie [...] (NRC, October 10th 1994)

English: Minister of foreign affairs Warren Christopher said in front of the American television [...]

These examples show that we also find patterns which often yield correct answers, but in fact are not inherently correct. In sentences such as 'Radio Free Europe was opened yesterday in Prague by the Czech president Havel' and 'The International Conference on Population and Development which was opened this morning in Cairo by the Egyptian president Mubarak [...]' the place names are not necessarily the capitals of the relevant countries. A conference (as well as an radio station) can very well be located in a city other than the capital. In this sense it is just a coincidence that the relation between the city and the country is a capital relation.

In other words, the sentence does not support the answer. For QA a supporting sentence is to be preferred. In TREC and CLEF such answers are typically evaluated as unsupported answers rather than correct answers. In the end, we will also need linguistic (that is semantic and syntactic) information besides frequency information to answer questions correctly.

5.6 Conclusion

In this chapter we automatically learn patterns to extract facts for off-line question answering by applying a bootstrapping technique. A crucial issue for existing bootstrapping-based algorithms is to ascertain that the patterns and instances found are reliable so as to avoid the extraction of too much noise during further iterations. It seems natural to focus on precision when working with bootstrapping techniques. Many unreliable patterns will result in noisy sets of instances, which in return will yield wrong patterns etc. However, storing the facts off-line lets us use frequency counts to filter out incorrect facts afterwards, therefore we do not need to focus on precision during the extraction process. It is of greater importance that we extract at least the correct answer.

We presented a bootstrapping algorithm for finding dependency-based patterns and extracting answers. The algorithm takes as input ten seed facts of a particular question category. On the basis of these ten seed pairs patterns are learned. We use these patterns to find new facts and these new facts can again be used to deduce patterns.

Experiments were performed on three different question categories: capital, football, and minister. For the capital and football categories the results showed indeed

that runs with a high recall and low precision achieved the best performance for QA, although for the capital category low precision was more harmful than for the football category. For the minister category one highly precise pattern was on its own responsible for the extraction of most of the facts, so the results for all experiments in this category were the same. We conclude that the results of the experiments suggest that for the benefit of off-line QA we better not focus on high precision and that recall is at least equally important. However, the balance between precision and recall can differ over different categories, depending on the kind of patterns discovered.