Mining Natural Language Answers from the Web

Günter Neumann and Feiyu Xu[1]
Language Technology Lab, DFKI
D-66123 Saarbrücken, Germany
{neumann, feiyu}@dfki.de

Abstract

We present a novel method for mining textual answers in Web pages using semi-structured NL questions and Google for initial document retrieval. We exploit the redundancy on the Web by weighting all identified named entities (NEs) found in the relevant document set based on their occurrences and distributions. The ranked NEs are used as our primary anchors for document indexing, paragraph selection, and answer identification. The latter is dependent on two factors: the overlap of terms at different levels (e.g., tokens and named entities) between queries and sentences, and the relevance of identified NEs corresponding to the expected answer type. The set of answer candidates is further subdivided into ranked equivalent classes from which the final answer is selected. The system has been evaluated using question-answer pairs extracted from a popular German quiz book.

Correspondence address:
Dr. Günter Neumann
Language Technology Lab, DFKI
D-66123 Saarbrücken, Germany
Phone: +49 681 302 5298
Fax: +49 681 302 5338
neumann@dfki.de

[1]The work presented in this article was supported by a research grant from the German Federal Ministry of Education, Science, Research and Technology (BMBF) to the DFKI projects Whiteboard (FKZ 01 IW 002) and Quetal (FKZ 01 IW C02). Thanks to Olga Goldmann for her implementation support. Also thanks to Ulrich Schäfer for providing us the Java interface to SMES-SPPC. The author's names are alphabetically ordered.

1. Introduction

Recent advances in the area of information extraction (IE) have demonstrated successfully that domain-specific processing of large free text collections is achievable by using shallow Natural Language Processing (NLP) components; it has been shown that shallow technology can provide sufficient information for highly accurate and useful tasks. The success in IE as well as in the area of Information Retrieval (IR) also have led to a very recent development of open-domain textual question answering systems (abbreviated as TextQA). They combine IR and IE technology in order to handle user queries of the sort *"Where is the Taj Mahal?*" by producing answer strings extracted from relevant NL paragraphs, e.g., "… *the Taj Mahal* ***in India*** …" where the found exact answer string is boldfaced. The majority of TextQA handle a limited sort of NL questions, namely factoid short-answer questions, where answers correspond to slot-oriented entities (e.g., when, where, who); beside single-valued questions, current systems also have started the exploration of list-based questions – e.g., *Which countries have a common border with Germany?* - and definition-based questions – e.g., *What is a battery?,* cf. [27]. TextQA systems are evaluated systematically as part of the Text Retrieval Conference series (TREC). Evaluation is performed by means of a large corpus of textual documents from which a set of question-answer pairs is mainly pre-selected as reference material. Very recently, the first evaluation of cross-language QA systems has been performed as part of the Cross Language Evaluation Forum (CLEF), where TextQA systems are evaluated which receive a NL query in one language (say German) and which returns answers for that query in textual documents written in another language (say English), cf. [23].

Currently, there is also an increased interest to consider the Web as an attractive resource for seeking answers for factoid questions in order to explore *open-domain web-based* question answering systems (abbreviated as WebQA), e.g. [15], [25], and [31]. Although the basic functionality between TextQA and WebQA is similar, there are some differences that make the development of WebQA a worthwhile venture. One important aspect of the Web is that of *data or answer redundancy*: the more frequently an answer appears, the easier it is to find it (cf. [5], [8], [16], [17]). The core assumption here is that with the massive amount of data on the Web it is more likely that a QA-system can find statements that answer the question in an obvious way, using simple pattern matching-based NLP methods (cf. [6], [8]). For example, if we consider a question like "*Who won the Nobel prize 2000 in chemistry?*", then a WebQA would consider all Web pages that contain the content words "won", "Nobel prize", "chemistry" and the time expression "2000" and any person name. The candidate answer person name could then be the person name with relevant distance to the other terms and the one with the highest likelihood. In this case the redundancy of the Web is used as a substitute for a deeper structural analysis that would probably be more successful in case of small data sources and hence less redundancy (cf. also [8]).

Another important aspect of the Web concerns multi-linguality [29]. The growth of multilingual users and content on the Web in recent years is impressive. According to the newest data in 2003 provided by Global Internet Statistics (cf. http://global-reach.biz/globstats/index.php3) , the English-speaking users are 35.6% (36.5% in year 2002), while the non-English users play now a dominant role with 64.4% (63.5% in 2002), in which the European languages contain 34.9% (35.5% in 2002) where German still is increasing (7.0% in 2003, 6.7% in 2002). At the same time, the content distribution of non-English Web pages has increased too: there are 68.4% English Web pages, among other languages, the distribution of German Web pages is at the third position with 5.8% following the Japanese pages (5.9%).
In this paper we describe WAG, an experimental search engine for performing automatic answer extraction from Web pages, allowing semi-structured queries for asking (e.g., who, when, where). WAG uses Google for performing an initial Web search. The N-best Web pages found by Google are linguistically processed by a shallow NL processor, which among others recognizes named entities (NE, such as person, company, location names, or time and date expressions) and maps each document into a stream of sentences. The main contributions of our approach for an open-domain Web-based answer extraction strategy are:

- A NE-directed voting technique by ranking all found NEs (independently from the fact whether they are relevant for the answer) using its term and document frequency, with
- an answer extraction and ranking strategy using word/NE overlap between query expression and answer candidates as scoring function, extending the approach of [16].

The answer extraction process as a whole realizes a kind of *text zooming* method in the sense that we first identify the relevant paragraphs, then the relevant sentences, and then the relevant NE, i.e., the exact answer. In all of the subsequent steps NEs serve as anchors for the selection of the relevant textual window. In doing so, the WAG-approach cannot only be used for extracting factoid answers, but can also be scaled-up for handling list-based and even template-based questions (cf. sec. 5). So far, the WAG system has been implemented and evaluated for German Web pages. However, the basic query and answer extraction strategies exploited by WAG are language independent, and have already served as a basic means for realizing a cross-language QA-system for German/English, cf. [21].

2. System Description

Our scientific view on the development of a generic question-answering system is that of a scalable and an adaptive system architecture. The idea is that depending on the complexity of the query information (from simple fact-based questions, to relational template-based questions, to thematic-oriented questions, see also [7]). Shallow or deep QA strategies should be selected (or even mixed) which might involve different degrees of linguistic processing, domain reasoning or interactivity between a user and the system.
In case of using the Web as answering source for factoid-based open domain questions, our current system design is focused on data directness, robustness, and scalability. For that reason we are using a simple query formulation process and few shallow NLP components. In the future we will scale up the system by integrating more advanced NLP components (as described in [22]).

2.1. Overview

Figure 1 displays the current system architecture of WAG. We will now describe briefly the major steps.

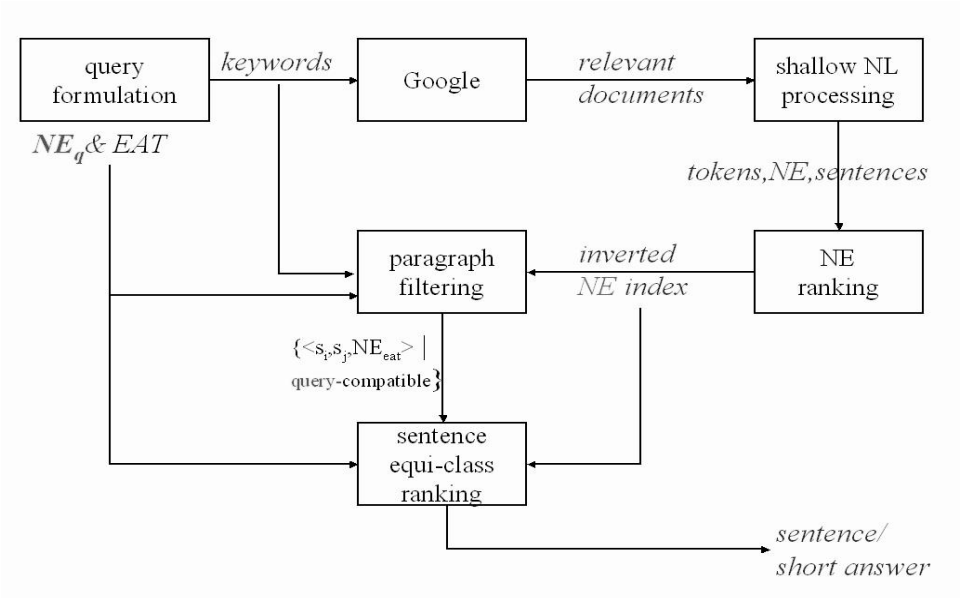

Figure 1 WAG's system architecture

Query Formulation. We are using a semi-structured template-oriented query formulation. The internal representation of an NL question consists of the set of relevant content words, named entities and the expected answer type (EAT). Thus for a question like "Who won the Nobel prize 2000 in chemistry?" the internal query object is as follows:

$$\left\{\begin{bmatrix} T: & tok \\ V: & \text{won} \end{bmatrix}, \begin{bmatrix} T: & tok \\ V: & \text{NobelPrize} \end{bmatrix}, \begin{bmatrix} T: & DNE \\ V: & 2000 \end{bmatrix}, \begin{bmatrix} T: & tok \\ V: & \text{chemistry} \end{bmatrix}, \begin{bmatrix} T: & PNE \\ V: & ? \end{bmatrix}\right\}$$

We denote such a representation as *Bag-of-Objects*. Here, an object is represented in form of a simple feature value structure, where *T* denotes the type of the object (either token (*tok*) or one of the types of the covered named entities, e.g., date expression (*DNE*) or person name (*PNE*)). The feature *V* holds the word form or phrase of an object. The object of the expected answer type EAT is represented by the NE in question and by the special symbol "?" for the value cell of the feature *V*. Note that it is possible to denote more than one expected answer type object, which can be used for processing of multi-factoid questions (e.g., *When and where was Mozart born*?, see also sec. 5).

In our current system the mapping from an NL question to such an internal query object is actually bypassed. Instead of parsing an NL expression, the user fills out a simple form that more or less corresponds to the structure of such an internal query object, e.g., the user enters the set of relevant content words, and selects named entities from a pop-up menu and specifies desired values (of course, freely selectable and including "?"). An additional motivation for this kind of query formulation is

that it also supports a transparent adaptation of the system to additional or new types of named entities or even domain specific ontologies (see also [18]).

Document Retrieval and Shallow NL Processing. All content words (all values from *V*) are passed to the Google search engine and the N-best documents (currently, N=50 is used) are further processed by our robust NLP system SMES-SPPC, cf. [22]. SMES-SPPC is a high-performance system for intelligent extraction of structured data from free text documents, especially designed for handling the German language.
German is a language with free word order, where morpho-syntactic information is coded via a rich inflectional system (e.g., different case and tense forms). Additionally, German has a very productive way of creating new words by means of compounding, including named entities (e.g., "Siemensgeneralvertreter'", *the general representative of Siemens*) and hyphen-coordination (e.g., "An- und Verkauf', *purchase and sale*). In case of a syntactic analysis, the German language allows for nearly arbitrary ordering of phrases and the splitting of verb groups into separate parts into which other phrases might be spliced in. To account for these challenging language properties, SMES-SPPC consists of a number of highly modular components for lexical processing (Part-of-Speech (PoS) tagging, morphological analysis, online compound analysis), named entity (NE) recognition, chunk parsing, and recognition of underspecified dependency structures of clauses. The system is very fast and can process about 12,000 words per second on the lexical level including NE recognition, and about 3000 words when considering also clause level processing on a standard PC (for more details see [22]).

Recognition of Named Entities. In WAG we are currently making use of the lexical components (which also recognize sentence markers) and the NE-recognition component. Since NE-recognition plays a crucial role for the paragraph selection strategies and answer extraction methods in WAG, we will describe some important aspects of our NE-recognizer.
The NE-recognition component currently handles 12 different generic NE types including names for persons, companies, locations and complex time, date and currency expressions, like "von knapp neun Milliarden auf über 43 Milliarden Spanische Pesetas" (*from almost nine billions to more than 43 billion Spanish pesetas*). The coverage is state-of-the-art, e.g., 93%/80% precision and recall for company names, 92%/90% for person names, and 98%/86% for locations. Each type of NE is defined as a finite-state sub-grammar that takes into account the specific context an NE appears in (e.g., company designator, PoS, morpho-syntactic properties of contextual elements). We are following a rule-based pattern-recognition approach similar to the approaches described in [2] and [13], such that NEs are identified basically by looking for contextual cues relying only on a small amount of NE-specific dictionaries (e.g., a small list of the names of the largest international companies). The major reason for doing this is that the creation and use of NE's change dynamically over time – which is especially true for the Web – so that a pure dictionary approach is not realistic. We also decided to use a rule-based approach, because statistics-based approaches (e.g., [4] and [3]) are still very corpus sensitive, and a rule-based approach is usually easier to maintain.[2]
Nevertheless, we observed that subsequent occurrences of already recognized NE's frequently appear in abbreviated form (e.g., "Siemens GmbH" and "Siemens"), often by making use only of a single

[2]However, currently a number of interesting *unsupervised* NE learning approaches have been developed which are interesting alternatives for a purely rule-based approaches, cf. [9] or [10], as well as [19].

word. Instead of defining specific recognition rules for these cases, we developed a method for the *online creation* of a dynamic NE lexicon. (Similarly, we are automatically expanding and memoizing unknown abbreviations). Once an NE has been recognized by means of the known rules (e.g., “Martin Marietta Corp.”), we store all words (without the contextual cues) in a lexicon (e.g., separate, but connected entries for “Martin Marietta”, “Martin” and “Marietta”). Then, for each unknown word sequence or known common noun (e.g., the word “March” is usually used to refer to the month, but could also be part of an NE, similarly “GATES” in “BILL GATES” if we ignore capitalization) that occurs in a certain distance to a recognized NE, we look it up in the dynamic lexicon. In this way, an NE-specific kind of co-reference resolution is performed.

2.2. Mining and Extracting Answers

Finding the answer of a simple fact-based query basically means finding a single named entity – an instance of the expected answer type of the question, e.g., a person name for a who-question. We assume that a single sentence will contain the answer. However, since the Web pages returned by Google and further processed by the NLP system will contain many, many NE expressions in a single Web page as well as in *multiple* Web pages, simply iterating through all sentences to look for candidate answers is not appropriate.
In order to take advantage of the redundancy of NE expressions, we compute a weight for each recognized NE term. In later steps, we use this list of weighted NE terms as anchor for the retrieval of relevant paragraphs and sentences, and for the ranking of candidate answers.

NE ranking. The rank r_{NE} of a named entity NE (and hence its relevance) is computed as follows:

$$(1)\ r_{NE} = \left|DL_{NE}\right| \times \left(\alpha * TF_{NE}\right) + \sum_{i=1}^{\left|DL_{NE}\right|} \left(1 - \frac{r_i}{N}\right)$$

where

- DL_{NE} is a list of documents containing the NE and ordered according to Google’s ranking,
- TF_{NE} is the frequency of the NE considering all relevant documents,
- r_i is the Google-rank of the ith document in DL, and
- α is a smoothing factor.

This means that an NE that occurs in more different documents will receive a higher weight than an NE that occurs in fewer documents (redundancy factor). Furthermore, NEs that occur in documents ranked higher by Google will receive a larger weight than NEs occurring in lower ranked documents (authority factor). It might be the case that an NE is expressed using different forms, for example, ”Heeger”, ”Alan Heeger”, ”Alan J. Heeger”. We are able to recognize these variants and treat them as synonyms before the NE-ranking is computed using the NE-reference resolution method of SMES-SPPC as outlined in the previous subsection.

Paragraph Selection. Once the rank for every recognized NE is computed, we construct an inverted index from the individual NEs to their positions in the original Web page. We further subdivide these indexed NEs by collecting all NEs of the same type into an individual list (e.g., a list of all found

person names). These NE-lists are used for paragraph selection, which works as follows: for each named entity from the NE-list which is *type-compatible* with the expected answer type of the current question (e.g., person) we determine each of its position P_{NE} in the original Web pages and extract an P_{NE}-centered window $S_1S_2S_{NE}S_3S_4$, where S_{NE} is the sentence containing the NE, and S_i are the adjacent sentences. For each triple $S_1S_2S_{NE}$, $S_2S_{NE}S_3$ and $S_{NE}S_3S_4$, a weight is computed based on the number of containing content words of the question and the distance of each identified content word from P_{NE}. The highest scored triple is selected as a candidate paragraph.
The major reason for computing a candidate paragraph first instead of directly returning S_{NE} is that of increased robustness: it might be that the sentence S_i which contains the highest overlap with the question does not contain an instance of the expected answer type, but a generic phrase (e.g., in case of referential expressions), then we are still able to consider S_i for the answer extraction process (see next section). In a similar way, we are able to handle named entities which have not been recognized by our NE-finder, but which co-occur with NE.

Sentence Ranking. Note that we consider each occurrence of a named entity NE in the selected document set. Redundancy comes into play here, because it might be that different paragraphs from different documents which contain NE, differ only in view wordings because they contain the same or very similar sentences. We now describe, how we can collapse similar sentences occurring in multiple documents into a single equivalence class. These classes are then used as the basis for selecting and ranking answer candidates. The core idea is to collect all sentences that have the same rank into one group. We denote such a group as a sentence-based equivalence class. In principle, the rank is determined by computing the overlap of tokens and NEs between the query and each sentence. The scoring function *EC* used in our approach for building and ranking sentence-based equivalence classes is an extension of the one described in [16]. *EC* is defined as follows:

$$(2)\quad EC = (olToken + olNE) * (1.5 + \frac{\sum_{i=1}^{n} r_{NE(EAT)_i}}{n})$$

Thus, we consider both the number of overlapped word forms or tokens (denoted as *olToken*), and the number of overlapped named entities (*olNE*), cf. also sec. 2.2. Note that this cannot include named entities that have the same type as the expected answer type (EAT) because the EAT actually serves as a typed variable in the question. However, we consider the weight of each NE that is compatible with the expected answer type (denoted as $r_{NE(EAT)}$). For n instances of the expected answer type occurring in a sentence (which means that the sentence has ambiguous answers), we use the average weight of the ambiguous NE's. In summary, a sentence has higher relevance than another one, if it shares more common words and named entities with the question, and if it contains instances of the expected answer type with high weights. The EAT compatible NEs are stored in a list and associated with the sentence equivalence class. These are chosen as exact answers ranked according to their weights. For example, for the question

(3) *Welches Pseudonym nahm Norma Jean Baker an?*
Which pseudonym did Norma Jean Baker use?

WAG returns a list of equivalence classes of sentences, and associated list of instances of ranked answer type named entities. (4) shows one equivalence class (abbreviated as eclass) for question (3). It contains only one sentence candidate, where two ranked person names ("Marilyn Monroe" and "Norma Jean Mortenson") can be potential exact answers. The person name with highest weight is "Marilyn Monroe", which is selected as the best exact answer in this case.

(4)
```
<eclass rank='7.254032855348923'>
<sentence url='http://www.beatlesfan.de/alle.htm'>
    Marilyn Monroe war der Kuenstlername von Norma Jean Mortenson, auch bekannt als
    Norma Jean Baker
    Marilyn Monroe was the stage name of Norma Jean Mortenson, also known as Norma
    Jean Baker
</sentence>
  <exact-answer  type ='PERSON'>
    <name rank='0.6029282321968642'>
        Marilyn Monroe
    </name>
    <name rank='0.024088195477597402'>
        Norma Jean Mortenson
    </name>
  </exact-answer>
</eclass>
```

In other words, this means that a sentence-based equivalence class performs a further "zooming" step such that it collects from all documents those sentences together which have the highest degree of similarity wrt. the query where similarity is determined by the scoring function *EC*. The sentences of a class are then used to construct the ranked list of EAT compatible named entities *NE(EAT)*. These then serve as the corresponding question's answer candidates from which the k-best are chosen and presented to the user. In some other sense, the process of determining the ranked list of exact answers can also be interpreted as a specific kind of *merging* of those NE's that are EAT-compatible and occur in similar sentences (of multiple documents). Thus our approach can also be used for handling list-based questions of the sort *Name 22 cities that have a subway system.*

Implementation. WAG has fully been implemented using Java and XML. SMES-SPPC is implemented in C++. Although WAG is still a prototype under development, the performance is encouraging. Running on a Linux-based Web server (4GB main memory), processing of a complete question-answer cycle ranges from around 20 to 120 seconds mainly depending on the time needed for harvesting the Web pages (however, no systematic performance tests have yet been done).

3. Evaluation

3.1. Starting Points

The WAG system has been evaluated by using question-answer pairs from a German popular quiz book written by [30]. The Quiz book covers questions from more than 40 subject areas, such as, philosophy, nature science, history, geography, sport, culture and religions, asking about gods, saints, investors, politicians, musicians, prize-winners, as well as, territories, lakes, churches and music bands, etc. Since we assume that these questions are formulized without any electronic QA system in mind, this test suite can be considered as a realistic scenario for evaluating WAG. A quiz question is paired with four multiple-choice answers in the Quiz book as shown in the following example (English translations are added by the authors):

In welchem Land gibt es das berühmte Loch Ness
(*In which country is there the famous Loch Ness*)?
A. Irland (*Ireland*) B. Schottland (*Scotland*)
C. Norwegen (*Norway*) D. Deutschland (*Germany*)

We use the answer provided by the book (for the example this would be B.) as the answer key for the evaluation and use them for performing two different experiments. Note that the answer keys corresponds to the exact answer provided by WAG and is also used as references for evaluating sentence answers.
In the first run (also denoted as **restricted test environment**), we have considered questions of two answer types: person and location, although our system can deal with other answer types, as well (e.g., date time, organization, number, address and currency). We have chosen the first 20 person-related and the first 19 location-related questions from the Quiz book, for which we had assumed that the NE-instances would be covered by our NE-recognizer. Here are some example question-answer pairs from the test corpus:

- Welches Pseudonym nahm Norma Jean Baker an? (person)
  Which pseudonym did Norma Jean Baker use?
- Wer wurde 1949 erster Ministerpräsident Israels? (person)
  Who became Israel's first prime minister in 1949?
- In welcher ehemaligen Sowjetrepublik befand sich das Kernkraftwerk Tschernobyl? (location)
  In which former soviet state was the nuclear power plant Chernobyl?
- In welcher europäischen Stadt nennt man die Altstadt Alfama? (location)
  In which European city is there an old city part called Alfama?

In the second run (denoted as **extended test environment**), we have added an additional question type, namely, *organization*, and have randomly chosen questions from the quiz book. This test corpus includes 25 location-related questions, 8 organization-related questions, and 35 person-related questions. The major difference wrt. the restricted test environment is that we do not know in advanced whether the expected answer type of a question is covered by our NE-recognizer, e.g., aid organization names, music band names, and multilingual names. Basically, our motivation for this sort of experiment was to test the performance of WAG, also in cases that the NE-recognizer is "out-of type". Illustrative examples are:

1. Wie heißt die 1971 gegründete Organisation, die Hilfe für alle in aller Welt leistet?
   What's the name of the organisation found in 1971, which helps people all over the world?

2. Welcher Boygroup gehörte der englische Sänger Robbie Williams vor seiner Solokarriere an?
   Which boy group did the English singer Robbie Williams belong to before his Solo Career?
3. Wie heißt der Europäer, der Ende des 15.Jahrhunderts Indien auf dem Seeweg erreichte?
   What's the name of the European, who arrived India via seaway?

The answer to the first question is "*Ärzte ohne Grenzen*" (*physicians without borders*). Linguistically, this is a complex noun phrase, which does not contain any proper name, while the answer to the second question is "*Take That*", which is very difficult to recognize as a named entity without knowing it as a band name beforehand. In order to be able to answer the third question, a multilingual person name recognizer would be needed, since "*Vasco da Gama*" is not a regular German person name (basically because of the *da* infix).
As the base case for our experiments we are using Google's snippets as answers. In our evaluation scenario, we choose the top 50 documents found by Google. For each processed question WAG returns fully automatically the following information:

1. The number of documents found by Google.
2. The top five Google snippets; we treat them as answers extracted by Google.
3. The top five sentence-based equivalence classes.
4. Each equivalence class contains the list of weighted named entities that are found instances of the expected answer type.
5. The time point when Google was called for that question; this is important because this can also help us to track changes in the answer results caused by dynamically changed Web content (when trying to answer the same question later).

Given these parameters, we evaluated the results based on the state of the art QA evaluation methods [27]. We consider two metrics: *recall* and *mean reciprocal rank* (MRR). Recall is the percentage of the questions answered correctly by WAG compared to all questions of the test corpus, cf. [15]. Thus if it contains N questions, then a recall of 0.5 means that WAG can answer half of them correctly, i.e., we do not consider "wrong answers". We consider two cases: the recall of the first relevant answer (top1) and the recall of the first three relevant answers (top3). MRR is the standard TREC effectiveness measure reported by NIST for each TREC QA run [28]. The value of MRR is the mean of the reciprocal values of the rank of all correct answers among the top N (in our current experiments we have chosen N=5):

$$MRR = \frac{\sum_{i=1}^{N} \frac{1}{rank_i}}{N}$$

In both experiments we consider the best five sentence equivalence classes, from which the first sentence and the first instance of the expected answer type is selected as sentence answer and exact answer, respectively.

3.2 Restricted Test Environment

Person Question Evaluation. In Table 1, Table 2 and Table 3, the average performance of Google snippets, WAG exact answers and WAG sentences is listed. In general, WAG has both a better recall and a better MRR value than Google, regarding both exact answers and sentences.

Google snippet	MRR (N=5)	Recall top1	Recall top3
all questions	0.103	0.3	0.35

Table 1: The average performance of the Google snippets (person questions)

WAG exact answer	MRR (N=5)	Recall top1	Recall top3
all questions	0.212	0.45	0.55

Table 2: The average performance of WAG exact answers (person questions)

WAG Sentence	MRR (N=5)	Recall top1	Recall top3
All questions	0,216	0.5	0.55

Table 3: The average performance of WAG sentence (person questions)

Location Question Evaluation. Table 4, Table 5 and Table 6 show that WAG has generally better performance than Google except for one case, namely the recall of top3.

Google snippet	MRR (N=5)	Recall top1	Recall top3
All questions	0,092	0.21	0.52

Table 4: The average performance of the Google snippets (location questions)

WAG exact answer	MRR (N=5)	Recall top1	Recall top3
All questions	0,135	0.31	0.42

Table 5: The average performance of WAG exact answers (location questions)

WAG sentence	MRR (N=5)	Recall top1	Recall top3
all questions	0,126	0.26	0.37

Table 6: The average performance of WAG sentences (location questions)

Evaluation Summary. The following table summarizes the performance of WAG vs. Google given the total 39 questions:

Metric	Exact answer (WAG)	Sentence (WAG)	Google Snippet
MRR (N=5)	0.174	0.171	0.0975
Recall top1	0.38	0.38	0.255
Recall top3	0.48	0.46	0.435

Table 7: The average values of all 39 questions

3.3. Extended Test Environment

For the extended test environment, we take only those questions into account for which Google returns results in form of documents.

Location Question Evaluation. In Table 8, Table 9 and Table 10, the average performance of Google snippets, WAG exact answers and WAG sentences is listed, given the 25 location questions. In general, WAG has both a better recall and a better MRR value than Google, regarding both exact answers and sentences, except that Google is slightly better concerning "Recall top3" than WAG exact answers.

Google snippet	MRR (N=5)	Recall top1	Recall top3
all questions	0.113	0.216	0.6

Table 8: The average performance of the Google snippets (location questions)

WAG exact answer	MRR (N=5)	Recall top1	Recall top3
all questions	0.16	0.44	0.56

Table 9: The average performance of WAG exact answers (location questions)

WAG Sentence	MRR (N=5)	Recall top1	Recall top3
all questions	0,183	0.32	0.72

Table 10: The average performance of WAG sentence (location questions)

Organization Question Evaluation. Eight organization questions are evaluated. Google and WAG sentence answers achieve comparable performance, as given in Table 11 and 13. Unfortunately, we yield a less good performance in the case of WAG's exact answers. The main reason is that most organization names cannot be detected by our NE-recognizer, which is due to the fact, that the NE-grammars for organizations mainly cover company names, and hence have quite restrictive "meaning", but the Quiz book seldom ask for company names.

Google snippet	MRR (N=5)	Recall top1	Recall top3
all questions	0,17	0.375	0.75

Table 11: The average performance of the Google snippets (organization questions)

WAG exact answer	MRR (N=5)	Recall top1	Recall Top3
all questions	0,09	0.125	0.375

Table 12: The average performance of WAG exact answers (organization questions)

WAG sentence	MRR (N=5)	Recall top1	Recall top3
all questions	0,16	0.375	0.625

Table 13: The average performance of WAG sentence answers (organization questions)

Person Question Evaluation. Although many multilingual names cannot be correctly identified by our NE-recognizer, our exact answers deliver similar performance as Google, as shown in Table 14 and 15. WAG sentence-answers has achieved best results, see Table 16.

Google snippet	MRR (N=5)	Recall Top1	Recall top3
all questions	0,126	0.34	0.48

Table 14: The average performance of the Google snippets (person questions)

WAG exact answer	MRR (N=5)	Recall top1	Recall Top3
all questions	0,127	0.314	0.4

Table 15: The average performance of WAG exact answers (person questions)

WAG sentence	MRR (N=5)	Recall top1	Recall Top3
all questions	0,154	0.4	0.485

Table 16: The average performance of WAG sentence answers (person questions)

Evaluation Summary. The following table summarizes the performance of WAG vs. Google Snippet given the total 68 questions:

Metric	exact answer (WAG)	Sentence (WAG)	Google Snippet
MRR (N=5)	0.126	0.166	0.136
Recall top1	0.29	0.365	0.31
Recall top3	0.445	0.61	0.61

Table 17: The average values of all 68 questions

Comparision with Other Systems. All published WebQA systems that are known to us are "English speaking" systems (e.g., START, [14]; Ionaut, [1]; MULDER, [15]; AnswerBus, [31]; NSIR, [25]). A direct comparison of the corresponding reported results with our result is difficult, because our evaluation data set contains only German Web pages and German questions. However, we feel that our system has yielded encouraging results (cf. Table 7 and 17). [31] has compared a number of systems using the TREC 8 corpus for evaluating sentence-level answer extraction. Relevant numbers he reports are (considering recall on "top1"): AnswerBus=60%, LCC[3]=37.5%, START=14.5%. We obtain a value of 38% (top1) for the restricted test, and 36.5% (top1) for the extended test.

[3] Cf. [8]

3.4. Qualitative Analysis

From a more qualitative point of view, the major performance influence of our methods can be summarized as follows (the positive effect of these issues has also been highlighted by [16]):

- Redundancy: NE-ranking,
- Relative score is better than absolute score: the integration of document ranking into NE-ranking and further the integration of NE-ranking into scoring function,
- Building equivalence classes based on scoring function for ranking the answer candidates,
- The ambiguity of exact answers is handled in the scoring function.

We will now discuss aspects of our qualitative analysis in more detail.

Does the Quiz Book provide actually valid answer keys? As mentioned above, we use the Quiz book as our gold-standard. However, during our evaluation, we found a trade-off between Quiz answers and WAG responses. For example, WAG returns "*Cambridge*" as a response to the question "*In which American city is the famous Harvard University located*?" although "*Boston*" is provided as the legitimate answer in the Quiz book. The sentence found by WAG in which our exact answer occurs, is:

> „Älteste und eine der angesehensten Hochschulen der USA ist die Harvard University in Cambridge, die 1636 gegründet wurde."
> *Oldest and one of the most famous Universities of USA is the Harvard University in Cambridge, which was founded in 1636.*

It turned out that our response is more precise than the Quiz answer, since Boston is a neighbor city of Cambridge and Harvard is in Cambridge. In general, a question can have more than one correct answer, but the Quiz book provides only one answer to each question. Consider, for example, the question "*With whom was Heinz Rühmann married*?". Here, the answer key in the Quiz book is "*Hertha Feiler*". However, we also found "*Maria Bernheim*" as his first wife, and "*Hertha Droemer*" as his last wife. In this case, we accept all these three names as correct answers. Thus, our experience approves the TREC evaluation approach ([27], [28]) that simply matching the exact answer against the answer key is insufficient and human judgement is needed in many cases.

Exact Answer and Sentence Answer. Evaluations of organization and person questions show that sentence-answer extraction is more robust than exact-answer extraction, since it is less dependent on the result of the NE-recognizer. However, Web pages are semi-structured entities, which may contain textual fragments with no "well-formed" sentence structure (at least at the surface level). Therefore, NL expressions extracted from those texts are often encoded in telegram formats or as part of HTML tables or similar structures. In these cases, an exact answer gives a more clear response to the question than such a semi-structured Web text, as long as the Web-page structure is not taken into account. For example, this is a possible answer to the question "*What is the name of the capital of Sachsen-Anhalt*?":

<eclass rank="5.96079367409883">
<sentence url="http://www.citypopulation.de/Deutschland-SachsenAnhalt_d.html">
Sachsen-Anhalt - Staedte und Provinzen - Einwohnerzahlen und Kartena = x if a! =x head d if a! =x show mapDeutschland-SachsenAnhalt, Sachsen-Anhalt, D, 1, 2, 2, 400, 450, yes Sachsen-Anhalt Hauptstadt: Magdeburg UebersichtRegierungsbezirke Die groessten Gemeinden Alphabetische Aufstellung der Gemeinden Regierungsbezirke NameAbk
</sentence>
<exact-answer type="LOCATION">
<name rank="1.0">magdeburg</name>
</exact-answer>
</eclass>

The sentence answer is an assembly of various tabular text fragments within the Web page. In principle, information wrapping technology would be needed in combination with shallow NLP in order to extract the right piece of text, namely, "*Sachen-Anhalt Hauptstadt: Magdeburg*", which we have yet not foreseen in our system. As a kind of compensation, the exact answer "Magdeburg" facilitates us to find the right span in the found "unwell-formed" sentence. On the other hand, a sentence answer usually provides informative context for an exact answer, which can help "understanding" and hence disambiguation of an exact answer by the user. Therefore, exact answers and sentence answers are complementary to each other with respect to system robustness and sufficiency of explanatory context for an answer.

Redundancy and Popular Information. Although redundancy can be generally utilized to overcome deep natural language understanding, majority polling alone is not sufficient to guarantee the correct answer. Therefore, in some cases the most popular information will be ranked higher than the correct answer. For example, "*Victoria Adams*" is the answer to the question "*With whom is the soccer idol David Beckham married*?". However, we detected that the most relevant documents are about the movie "*Kick it like Beckham*". Although the extracted sentences contain words "married", "soccer", "idol" and "David Beckham", they are about a girl in the film who adores Beckham as a soccer idol and marries a man, who is called "Beckham", too. We assume, that even in the case of WebQA a certain degree of syntactic parsing will be necessary to uncover relevant dependency relations. In this case the specification of a more sensitive similarity measure for sentences might be possible, which takes into account structural abstraction. However, since processing of Web data requires an extreme good performance (in case of robustness and efficiency), a careful selection of relevant paragraphs is necessary before the deeper linguistic analysis shall be applied.

Anaphora Resolution. The overlap strategy is useful to find sentences, which maximally match the questions. But these sentences contain not always the exact answers, for example, for the question "*In which city is the Church of the Nativity of Jesus Christ*?" the best ranked sentence is "In der knapp zehn Kilometer von Jerusalem entfernten Stadt steht heute dort, wo Jesus geboren worden sein soll, die Geburtskirche." (*In the city, which is about ten kilometres from Jerusalem, is the Church of the Nativity, where Jesus was born).* In this sentence, the anaphoric noun phrase "*the city*" refers to the city name "*Bethlehem*", which was mentioned in the preceding sentence: "Das Lukas-Evangelium nennt als Ort der Geburt Jesu Bethlehem beziehungsweise dessen Umgebung." (*The Gospels of Luke called the birth place of Jesus Bethlehem, respectively, the surrounding areas.)* In order to obtain the

exact answer, anaphoric resolution strategies are needed. In WAG, we have access to the paragraphs from which the answer candidate sentences have been extracted. These paragraphs can then be used as the discourse context for anaphoric resolution using shallow reference mechanism as, for example, described in [11], where we show how a shallow reference resolution strategy can be constructed on basis of the finite state approach of our shallow NL system SMES-SPPC.

4. Related work

Our experiments have approved the experiences reported in [5], [8], [14], [15], [16] and [17], that the redundancy plays an important role for WebQA. To best of our knowledge, WAG is the first publication of an open-domain "German speaking" WebQA. AnswerBus developed by [31] also allows German input queries. However the queries have to be translated (using the translator provided through Babelfish[4]) into English because answer extraction is only performed for English Web pages. AnswerBus only considers sentence level Web information retrieval. It uses the following formula to select sentences that may be an answer: $q \geq \lfloor \sqrt{(Q-1)} \rfloor + 1$, where Q is the number of words in a query, and q is the number of matching words in a retrieved sentence. This is similar to our overlapping approach. One major difference is that it does not take into account named entity extraction for sentence selection and that it cannot deliver exact answers.
[8] also makes explicit use of redundancy information for answer ranking. For 50 byte answers to the TREC-9 questions[5]), [8] shows that redundancy has a very crucial impact on boosting the performance of their system. Similarly to the NE-ranking measure in WAG, it weights candidate answer terms by combining a redundancy factor with a term frequency factor. The main differences are a) they count the passage frequency instead of the document frequency and b) they do not take into account the relevance of the document or passage given through the document or passage retrieval.
The WebQA system MULDER (cf. [15]) also chooses Google for its document retrieval task. In a first step, the answer extraction module extracts the matching text snippets (also called summaries), from which relevant phrases are extracted serving as exact answer candidates. The idea of clustering ranked answers is similar to our equivalence class method. They also collect name variations and provide them as alternative answers. The member with the highest score in each group is selected as representative for that group and is presented to the user as a potential answer. The major differences wrt. WAG are a) they do not make use of NE-ranking (simply because they do not use a NE-recognizer), and b) MULDER utilizes the inverse document frequency (IDF) together with word distance to extract summaries. These extracted answers are then assigned scores based on their contexts in documents and then further clustered into groups. Furthermore, MULDER already makes use of relative deep NLP components (e.g., syntactic parser for query *and* documents, word net), which influence the run-time performance negatively, as noted by [25]. Although we are using only very few NLP components, our current results are competitive.
NSIR ([25]) also makes use of shallow NLP and generic search engines, such as, AlltheWeb, Google and Northern Light for document retrieval. Instead of paragraph retrieval, they directly extract sentences, using N-gram and vector space models for the sentence-based ranking. Given the relevant sentences, they use a generate-and-filter strategy to find exact answers. They extract at first all phrases

---

[4] http://uk.altavista.com/babelfish

[5] For the 50 byte task, it suffices to determine a 50 byte long string which *contains* the exact answer. Thus, this task is in general simpler than the task of determining the exact answer string.

and then rank them by using similar heuristics, as we use in our paragraph retrieval and overlap strategy. For example, a phrase will receive a higher score than another one, if it contains more query words and/or are next to a large number of query words (a similar techniques is also described in [21]). NSIR does not use any named entity recognition tool for answer type identification. They map phrases to answer types by means of their internal part of speech information. Since, they do not use a NE-recognizer, they are more robust concerning identification of instances of the expected answer type. In contrast to our answer extraction approach, they consider all sentences, where we only consider small NE-anchored paragraphs.

In [6], the QA system AskMSR is described which also favors to start system development using simple data-oriented techniques first instead of beginning QA system design with rich linguistic resources. In order to do so, they also favor answer redundancy as an important measure for answer candidate selection and ranking. During query processing each NL query is mapped to a set of possible answer patterns. For example, for the query "When was Abraham Lincoln born?" a possible answer pattern is "Abraham Lincoln was born on <DATE>". These patterns are then passed to the search engine and the summaries (i.e., the snippets) are used for answer selection. In order to increase recall, they perform a re-formulation process on level of the surface patterns in order to retrieve documents, which contain similar instances. Such a process might be useful for languages like English, which has a restricted word ordering, and hence a small set of pattern variations might suffice. However, it might be *less appropriate* for handling free word order languages like German especially in case of processing large questions. Furthermore, AskMSR does not consider whole documents but document summaries for the identification of possible answer candidates. Thus it might be more difficult to scale up the system for handling more complex questions like list based questions, because for such kind of questions, answer candidates might be distributed across multiple documents.

In this context it is interesting to refer to the work of [26] and [19]. In [26] also the power of surface text patterns for open-domain QA systems is explored by presenting a method for automatically learning patterns of the form "<NAME> was born in <BIRTHDATE>" to be useable for answer candidate selection from the Web. The method they present uses a bootstrapping approach to build a large tagged corpus starting with only a few examples of QA pairs. In [19] a set of algorithms is presented for distinguishing personal names with multiple reading, e.g., "Jim Clark – Race car driver from Scotland; Jim Clark – Netscape Founder". The approach utilizes an unsupervised clustering technique over a rich feature space of biographic facts, which are automatically extracted via a language-independent bootstrapping process. Integration of variants of these learning mechanism into a WebQA like ours, for example, would help to decrease the error rate in the answer selection phase because it could help to further restrict the set of possible answer candidates.

Although not directly comparable to current WebQA systems, there are ongoing interesting works in the area of database query formation from natural language, which one could also denote as DatabaseQA. Of most interest in the context of our current work are the methods described in [20] and [24]. In [20] a method is described that performs NL analysis on a very shallow level by means of keyword-based triggering of parts of the database graph and a statistical keyword based disambiguation. In [24] a similar technique is described which uses some more NL preprocessing, but which also uses a robust and shallow triggering of relevant parts of the database graph. The most interesting aspect of the approach described in [24] is the notation of "reliable NL interface", which provides a means for the DatabaseQA for distinguishing "simple" and "hard" questions. The major obstacle of both approaches is that they rely on a closed-world assumption wrt. the content of the

database graph. However, it might be worthwhile to explore the application of these approaches for the modeling of hybrid QA systems, i.e., QA systems that can handle different sorts of data pools using a uniform NL interface.

5. Conclusion and future work

We described and evaluated WAG, a Web-based answer extraction system for extracting sentence answers and short answers from German Web pages automatically using semi-structured fact-based queries. WAG uses ranking of named entities and statistical based text zooming as major strategies. It can handle ambiguous answers and performs a multi-document answer extraction process.
The focus of our research so far is based on simplicity and robustness following a data-driven, bottom-up system design. Of course there is enough room for increasing the performance of WAG. For example, currently we do no query expansion or re-formulation which would surely help to increase at least the recall. Furthermore, our semi-structured query format allows us very easily to express multi-fact (or template-based) questions (by specifying more than one expected answer type of different type). This could also be viewed as specifying a kind of on demand template (who did what when where). In this case our paragraph selection and sentence ranking process would be extended to return partially filled templates that have to be merged in a later step to find candidate templates. In some sense, the whole approach performs a kind of “template mining”. We have already implemented a first prototype, however not yet evaluated.
A further line of research that we are considering is the development of hybrid QA systems, e.g., by providing a uniform NL interface for TextQA, WebQA, and DatabaseQA. In [21] we have already extended our current approach for performing cross-language TextQA. We are now trying to extend our approach for performing database retrieval along the line of [20] and [24], since it seems that our internal query/document representation already is quite similar to the formalism introduced in [24].

References

[1] Steven Abney, Michael Collins, and Amit Singhal,
Answer Extraction,
in: Proc. the Conference on Applied Natural Language Processing, Seattle, Washington, 2000.

[2] Douglas E. Appelt, Jerry R. Hobbs, John Bear, David Israel and Mabry Tyson,
FASTUS: A Finite-State Processor for Information Extraction from Real-World Text,
in: Proc. the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93), 1993.

[3] D. Bickel, S. Miller, R. Schwartz, and R. Weischedel,
Nybmle: A high-performance learning name-finder,
in: Proc. the 5th ANLP, Washington, 1997.

[4] Andrew Borthwick,
Survey Paper on Statistical Language Modelling,
Technical Report, New York University, 1997.

[5] Eric Breck, Marc Light, Gideon S. Mann, Ellen Riloff, Brianne Brown, Pranav Anand, Mats Rooth, and Miachael Thielen,
Looking under the hood: Tools for diagnosing your question answering engine,
in: Proc. the ACL-01 Workshop on Open-Domain Question Answering, 2001.

[6] Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng,
Data-intensive question answering,
in: Proc. the Tenth Text REtrieval Conference (TREC 2001), 2001.

[7] J. Burger, C. Cardie, V. Chaudhri, R. Gaizauskas, S.Harabagiu, D. Israel, C. Jacquemin, C. Lin, S. Maiorano, G. Miller, D. Moldovan, B. Ogden, J. Prager, E. Riloff, A. Singhal, R. Shrihari, T. Strzalkowski, E. Voorhees and R. Weischedel,
Issues, Tasks and Program Structures to Roadmap Research in Question & Answering (Q&A),
in: NIST DUC Vision and Roadmap Documents, http://www-nlpir.nist.gov/projects/duc/roadmapping.html, 2002.

[8] Charles L. A. Clarke, Gordon V. Cormack and Thomas R. Lynam,
Exploiting Redundancy in Question Answering,
in: Proc. the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New Orleans, September, 2001.

[9] M. Collins and Y. Singer,
Unsupervised Models for named entity Classification,
in: Proc. EMNLP, 1999.

[10] S. Cucerzan and D. Yarowsky,
Language Independent NER using a Unified Model of Internal and Contextual Evidence,
in: Proc. the 6th CoNLL, 2002.

[11] Thierry Declerck and Günter Neumann,
A Cascaded Shallow Approach to Reference Resolution,
in: Proc. the EuroConference on Recent Advances in NLP (RANLP'01), September 5-7, 2001, Tzigov Chark, Bulgaria.

[12] Sanda Harabagiu, Dan Moldovan, Marius Pasca, Mihai Surdeanu, Rada Mihalcea, Roxana Gîrju, Vasile Rus, Finley Lăcătuşu, Paul Morărescu and Răzvan Bunescu,
Answering complex, list and context questions with LCC's Question-Answering Server,
in: Proc. the TREC-10, 2001.

[13] Ralph Grishman,
The NYU System for MUC-6 or Where's the Syntax**,**
in: Proc. the 6th Message Understanding Conference (MUC-6), 1995.

[14] Boris Katz,
From Sentence Processing to Information Access on the World Wide Web,
AAAI Spring Symposium on Natural Language Processing for the World Wide Web, Stanford, California. 1997.

[15] Cody Kwok, Oren Etzioni, and Daniel S. Weld,
Scaling question answering on the Web,
in: Proc. the 10th International World Wide Web Conference (WWW10), 2001.

[16] Marc Light, Gidcon S. Mann, Ellen Riloff and Eric Breck,
Analyses for Elucidating Current Question Answering Technology,
Natural Language Engineering 1(1), Cambridge University Press, 2001.

[17] Jimmy Lin,
The Web as a Resource for Question Answering: Perspective and Challenges,
in: Proc. LREC 2002, Spain, 2002.

[18] A. Maedche, G. Neumann and S. Staab,
Bootstrapping an Ontology-based Information Extraction System,

in: Intelligent Exploration of the Web, Szczepaniak, Piotr S.; Segovia, Javier; Kacprzyk, Janusz; Zadeh, Lotfi A., eds., Springer, ISBN 3-7908-1529-2, 2002, pp. 345-360.

[19] Gideon S. Mann and David Yarowsky,
Unsupervised Personal Name Disambiguation,
in: Proc. the 7th CoNLL, 2003.

[20] Frank Meng and Wesley Chu,
Database Query Formation from Natural Language using Semantic Modelling and Statistical Keyword Meaning Disambiguation,
Technical Report CSD-TR 990003, University of California, 1999.

[21] G. Neumann and B. Sacaleanu,
A Cross-language Question/Answering System for German and English,
in: Working Notes for the CLEF 2003 Workshop, Carol Peters ed., Trondheim, Norway, 2003.

[22] G. Neumann and J. Piskorski,
A Shallow Text Processing Core Engine.
in: Computational Intelligence, vol. 18, no. 3, August 2002, pp. 451-476.

[23] Carol Peters ed.,
Working Notes for the CLEF 2003 Workshop, online proceedings,
http://clef.iei.pi.cnr.it:2002/2003/WN_web/-working_notes_CLEF2003.htm,
Trondheim, Norway, 21-22 August, 2003.

[24] Ana-Maria Popescu, Oren Etzioni and Henry Kautz,
Towards a Theory of Natural Language Interfaces to Databases,
in: Proc. the International Conference on Intelligent User Interfaces, Miami, FL, USA, 2003.

[25] Dragomir Radev, Weiguo Fan, Hong Qi, Harris Wu and Amardeep Grewal,
Probabilistic question answering on the web,
In: Proc. the Eleventh International World Wide Web Conference, 2002.

[26] Deepak Ravichandran and Eduard Hovy,
Learning Surface Text Patterns for a Question Answering System,
in: Proc. the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002), 2002.

[27] Ellen M. Vorhees,
Overview of the TREC 2002 Question Answering Track,
in: Online Proc. the Eleventh Text Retrieval Conference (TREC 2002),
http://trec.nist.gov/pubs/ trec11/t11_proceedings.html, 2002.

[28] Ellen M. Voorhees and Dawn Tice,
Building a question answering test collection,
in: Proc. 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Athens, August 2000, pp. 200-207.

[29] Feiyu Xu,
Multilingual WWW – Modern Multilingual and Cross-lingual Information Access Technologies,
in: Knowledge-Based Information Retrieval and Filtering from the Web, Witold Abramowicz, ed., Kluwer Academic Publishers, chapter 9, 2003, pp. 165-184.

[30] René Zey,
Quiz für Millionen,
Falken Verlag. Niedernhausen, 2001.

[31] Zhiping Zheng,
AnswerBus Question Answering System,
in: Proc. HLT Human Language Technology Conference (HLT 2002), San Diego, CA. March 24-27, 2002.