Language Independent Answer Prediction from the Web

Alejandro Figueroa and Günter Neumann*

Deutsches Forschungszentrum für Künstliche Intelligenz - DFKI,
Stuhlsatzenhausweg 3, D - 66123, Saarbrücken, Germany
alejandro@coli.uni-sb.de, neumann@dfki.de

Abstract. This work presents a strategy that aims to extract and rank predicted answers from the web based on the eigenvalues of a specially designed matrix. This matrix models the strength of the syntactic relations between words by means of the frequency of their relative positions in sentences extracted from web snippets. We assess the rank of predicted answers by extracting answer candidates for three different kinds of questions. Due to the low dependence upon a particular language, we also apply our strategy to questions from four different languages: English, German, Spanish, and Portuguese.

1 Introduction

Normally, textual Question Answering (textQA) systems receive natural language queries as input, process large unstructured document collections, and return precise answers as output. The success of current textQA technology is due to the fact that they are combining technology from different areas (e.g., information retrieval, information extraction and natural language processing) in novel ways, cf. [1]. However, scaling this new QA technology to the Web in order to improve current search engines to efficiently locating information presents extraordinary challenges, cf. [2]. Consider for example the enormous size of the Web content that is currently indexed by the best search engines (Billions of Web pages). While an indexing of TREC–like corpus (only few Gigabytes in size, mainly newspaper text sources, fixed time period) on basis of NLP–oriented preprocessing has been shown to be very fruitful for textQA technology, doing the same for the Web is out of the reach with current technology. Another important aspect of the Web is its growing multilinguality, cf. [3]. Therefore, the exploration of language independent QA core technology is requested.

There exists first web–based QA systems (webQA) that successfully demonstrate how QA technology might improve future search engines by systematically exploiting the redundancy of the Web space, e.g., [4,6,3,5]. All of these systems have a similar architecture and perform three major steps:

* The work presented here was partially supported by a research grant from the German Federal Ministry of Education, Science, Research and Technology (BMBF) to the DFKI projects Quetal (FKZ: 01 IW C02) and HyLaP (FKZ: 01 IW F02).

T. Salakoski et al. (Eds.): FinTAL 2006, LNAI 4139, pp. 423–434, 2006.
© Springer-Verlag Berlin Heidelberg 2006

1. conversion of NL questions to search engine specific queries
2. interface to public search engines for document retrieval
3. extraction of answers from the retrieved web pages

The first step is needed in order to take advantage of a particular search engine's query syntax, and to increase the accuracy of potential relevant documents. The latter can be seen as a kind of "answer context prediction". For example a question like "Who is the president of Germany?" might be converted to search engine queries "the president of Germany is" or "Germany's president". Of course, without any corpus analysis this would just be a "blind" generate–and–test approach, so often these answering patterns are computed and weighted on basis of a statistical data analysis, cf. e.g., [6] and [7].

This sort of NL query analysis is similar to a query expansion strategy which is applied *before document retrieval*. In IR there exists also an alternative query expansion strategy, namely to perform the query expansion *after document retrieval*, which is also known as *pseudo relevance feedback* (PRF), cf. [8]. The advantage of PRF is that one can achieve a data–driven query expansion using the most relevant documents retrieved by the IR system. Applying this technology in a webQA system on basis of the retrieved documents, however might be negatively influenced by the time needed for fetching the N–best documents. Since this crawling process has to be done online, it defines a critical parameter for the latency of the subsequent webQA processes, and hence, might negatively effect the performance of the whole webQA system, cf. [6]. Fortunately, almost all modern search engines return a brief textual summary (called *snippet*) together with the URL of the candidate documents immediately as part of the query result.

In this paper, we apply the idea of PRF in the context of webQA, by performing query expansion on the set of N–best snippets retrieved using the NL query as it is, i.e., without converting it initially to possible query paraphrases. In contrast to the answer context prediction step mentioned above, we propose an *answer candidate prediction* strategy: the expanded query terms are interpreted as either direct answers to the NL user query or as terms which are semantically related to potential answer candidates. Note that answer candidates are usually not complete sentences but rather phrasal entities. Our answer predication strategy is completely data–driven. Thus it is very robust wrt. the form of the snippets and it is highly language independent. In order to evaluate our new methods we considered three specific type of questions for four languages (German, English, Spanish, Portuguese). We also designed answer extraction modules that resemble traditional systems, so to compare the extracted answers with the answers in the CLEF multilingual question/answer corpus.[1]

[1] For more information on CLEF (Cross–language Evaluation Forum) see http://www.clef-campaign.org/ .

2 System Overview

The user enters via some input device a NL query which is further passed to a search engine. The search engine returns a ranked list of document links together with a snippet for each document. The best N–snippets are passed to the answer prediction component. This component extracts from all snippets the best predicted answer strings. A predicted answer string is a substring extracted from the snippets for which a high semantic similarity to the question has been determined. Unlike PRF, the predicted answers are ranked and the M–best are submitted to the answer extraction component. It further splits the predicted answer strings into smaller units which might correspond to exact answer strings. The answer extraction component uses the NL user question in order to determine the expected answer type (EAT). For example, for a question like "Who is the president of USA?", the instance of EAT is PERSON. Since our goal is to be as language independent as possible and our focus is to evaluate the quality of the answer prediction strategy, this step resembles any traditional system based on pattern matching and lexical databases.

3 Ranking Scheme

Our system ranks two kind of strings: sentences and predicted answers. Since both are treated in the same way, we only describe the problem of ranking sentences in more detail. Formally, it can be specified as:

$$R = \{(s_1, l_1), (s_2, l_2), \ldots, (s_\sigma, l_\sigma)\}$$

where R is the rank of the set S of sentences of document D; s_s is the s–th sentence in S, $1 \leq s \leq \sigma$, where σ is the number of sentences in the document. We say that s_1 is preferred over s_2, if $l_1 > l_2$, where $l_s = rank(s_s)$, and $rank$ is a ranking rule that maps from the sentences to rank labels $rank : S \rightarrow L$.

3.1 Document Representation

In our system, a document is a multi–set of all the sentences which are extracted from all the N–best snippets returned by the search engine. We are using very simple rules for mapping a snippet to a stream of sentences, basically by using the standard punctuation signs as splitting points: colon, semicolon, coma, and dot. We will use W (the dictionary) for the set of all unique words in D, and $\omega = |W|$ the size of W. We start our description of a vector–space document representation by defining the following binary variable:

$$X_{sik} = \begin{cases} 1 \text{ if the word } w_i \text{ is in the sentence } s_s \text{ at position } k \\ 0 \text{ otherwise.} \end{cases}$$

Let $len(S_s)$ be a function which returns the number of words in a sentence S_s. Then, the frequency of the word w_i in the document is given by:

$$freq(w_i) = \sum_{s=1}^{\sigma} \sum_{k=1}^{len(s_s)} X_{sik}, \quad \forall w, \ 1 \leq i \leq \omega \tag{1}$$

Let w_j be a word in W, $1 \leq j \leq \omega$. For example, in a document D="JOHN LOVES MARY. JOHN KISSES MARY EVERY NIGHT.", we find two sentences determined by the dot. If we consider that "w_1" is "JOHN", then X_{111} will match the first occurrence of "JOHN" and X_{211} the second. X_{s1k} takes the value of one for only this two occurrences. Therefore, $freq$("JOHN") will be the sum of $X_{111} + X_{211} = 2$.

A document D is represented by the set of tuples:

$$D = \{\langle w_i, w_j, \epsilon, freq(w_i, w_j, \epsilon)\rangle,\ \forall\, i, j, \epsilon, 0 \leq \epsilon \leq \Upsilon\ \wedge\ freq(w_i, w_j, \epsilon) > 0\}$$

where $freq(w_i, w_j, \epsilon)$ is the frequency of w_i with which it appears to the left of w_j, Υ is the length of the longest sentence in the document, and ϵ is the absolute distance of their positions in the sentence:

$$freq(w_i, w_j, \epsilon) = \sum_{s=1}^{\sigma} \sum_{k=\epsilon+1}^{len(s_s)} X_{si(k-\epsilon)} X_{sjk} \tag{2}$$

For instance, $freq$("JOHN", "MARY", 1) = 2 means that the pattern JOHN * MARY was observed 2–times in document D. We also define $\Gamma(w_i, w_j, \epsilon, v)$: $W \times W \times N \times N \rightarrow \{0, 1\}$, as a function that returns 1 if the $freq(w_i, w_j, \epsilon)$ is equal to v, otherwise it returns zero. Using this notation, we define:

$$G(v) = \sum_{i=1}^{\omega} \sum_{j=1}^{\omega} \sum_{\epsilon=1}^{\Upsilon} \Gamma(w_i, w_j, \epsilon, v) \tag{3}$$

$G(v)$ determines the amount of pairs of words that occur v times in the document. In our example, the only tuple that occurs two times is JOHN * MARY, then $G(2) = 1$.

3.2 Ranking Sentences

We rank a sentence s_s in a document by means of a specially designed matrix M. This matrix is constructed from the tuples in D in the following way:

$$M_{ij}(s_s) = \begin{cases} freq(w_i, w_j, \epsilon) \text{ if } i < j; \\ freq(w_j, w_i, \epsilon) \text{ if } i > j; \\ 0 \qquad\qquad\quad \text{otherwise.} \end{cases}$$

w_i and w_j are two words in s_s, ϵ is the distance between w_i and w_j, ϵ=abs(i-j), $0 \leq \epsilon \leq \alpha$, and α=$len(s_s)$. This matrix models the strength of the relation or correlation between two words w_i and w_j in a sentence s_s.

The following filtering rule (which is the same for all languages) reduces the size of the representation of D and the noise of long sequences of low correlated words:

$$\forall i, j\ \ M_{ij} \leq \zeta \Rightarrow M_{ij} = 0$$

where ζ is an empirical determined threshold. This rule allows us to remove some syntactic relations of a word which are probably not important. For example, the English word *of* is a *closed class word* and as such will co-occur very often with different words at different positions. However, if it is part of a phrase like *The President of Germany*, the definition above allows us to keep *of* in the noun phrase, because it typically occurs with short distance in such specific syntactic construction.

Now, we define the **rank of a sentence** s_s as follows:

$$rank(s_s) = \lambda_{max}(M(s_s))$$

where $\lambda_{max}(M(s_s))$ is the **greatest eigenvalue** of the matrix M constructed from the sentence s_s, see also [14]. This eigenvalue gives us the amount of "energy" or "syntactic bonding force" captured by the eigenvector related with λ_{max}. Note that computing the eigenvalues for a small matrix is not a demanding task, and M is a matrix of size $len(s_s)$, which in case of snippets is small. There are two more aspects of M that is worths mentioning:

1. $\forall i\ M_{ii} = 0 \Rightarrow \sum_{\forall i} M_{ii} = 0 \Rightarrow \sum_{\forall f} \lambda_f = 0$.
2. $\forall i, j\ M_{ij} = M_{ji}$, the *spectral theorem* implies that $\forall f\ \lambda_f \in \Re$, and all the eigenvectors are orthogonal.[2]

The second aspect guarantees that for each sentence S_s, we will obtain a real value for $rank(s_s)$.

3.3 Extracting Predicted Answers

The matrix M contains the frequency of each pair of words of s_s, which appears in this sentence and which has the same distance in the whole document. We interpret sequences of word pairs which frequently co–occur with same distance in M as *chains of related words*, i.e., groups of words that have an important meaning in the document. This is important if we also consider the fact that, in general, snippets are not necessary contiguous pieces of texts, and usually are not syntactically well–formed paragraphs due to some intentionally introduced breaks (e.g., denoted by some dots betweens the text fragments). We claim that these chains can be used for extracting answer prediction candidates. Algorithm 1 extracts predicted answers from a sentence s_s. It aims to replace low correlated words with a star, where a low correlated word is a word in a sentence that has a low correlation with any other word in the same sentence. Sequences of high correlated words are separated by one or more stars. Thus, low correlated words in a sentences define the points for cutting a sentence into smaller units.

[2] The *spectral theorem* claims that for a real symmetric n-by-n matrix, like M, all its eigenvalues λ_f are real, and there exist n linearly eigenvectors e_f for this matrix which are mutually orthogonal.

Algorithm 1: extractPredictedAnswers

```
input : M,s_s
1  begin
2      predictedAnswers = s_s;
3      if numberOfWords(w_i) > 3 then
4          forall w_i ∈ s_s do
5              flag = true;
6              forall w_j ∈ s_s do
7                  if M_ijε > 0 then flag=false;
8              end
9              if flag then replace w_i with "*";
10         end
11         predictedAnswers = split(s_s,"*");
12     end
13     return predictedAnswers;
14 end
```

3.4 Ranking Predicted Answers

We rank every predicted answer ν extracted from a sentence s_s according to the following formula:

$$rank(\nu) = rank(s_s) * \sum_{b=2}^{\beta} P(B_b|B_{b-1})$$

where B_b are the words in ν, and β its length. This formula weights each piece of the sentence according to the probabilities of their bi–grams, which are estimated by the following formulae:

$$P(B_b|B_{b-1}) = \frac{log(freq(B_{b-1}, B_b, 1))}{log(freq(B_{b-1}))}$$

where we use the logarithm to smooth the frequencies, so to reduce the trend to favor high frequent words [9]. We consider the summation of the probabilities of bi–grams because we want to bias the ranking in such a way that longer predicted answers are preferred over shorter ones. Finally, redundant predicted answers are removed. A predicted answer ν is redundant if and only if the following conditions hold:

1. If there exists another predicted answer $\nu^{'}$, such that $rank(\nu) < rank(\nu^{'})$.
2. If ν is a substring of $\nu^{'}$.

If both conditions hold, we say that $\nu^{'}$ *contains* ν. For this comparison, we consider capitalized strings.

4 Answer Extraction

There is no standard strategy to evaluate predicted answers, but it is clear that the goal is to help the answer extraction step. Evaluating the predicted answers

in a straight forward way is too ambiguous and/or unfair. For this reason, we assume that extracting answer candidates from the rank of predicted answers gives us an unbiased notion of how good is the distribution of the predicted answers which do not contain an answer candidate. During this step, no further re-ranking is performed.

In general, a *correct answer* corresponds to an instance of a concept, which is the focus or the *expected answer type* (EAT) of a question, e.g., a person name for a Who–question. This information can then be used to locate possible instances in the predicted answer.

Table 1. Some sample Wh–question keywords for the covered languages

	Keywords
Date	Wann, When, Cuándo, Qué año, Welchem Jahr, Que ano
Location	Wo, Where, Dónde, Onde
Person	Wer, Who, Quién, Quem

Usually, a sophisticated Wh–question analysis is performed in order to extract the EAT and other important control information, cf. [1]. However, since we are interested in language independent techniques and how our strategy behaves in a traditional question answering system, we are making use of a very shallow strategy for the analysis of Wh–questions, which simply searches for some Wh–keywords (see Table 1) in the question in order to determine the EAT. The predicted answers are passed on to the corresponding answer extraction module, whose main task is to remove predicted answers that has no relation with the EAT. At this step, many good predicted answers are discarded. From the remaining candidates, answer candidates are extracted applying simple specialized extraction algorithms.

Currently, we only consider Who/Where/When–questions. These are also used in TREC and CLEF QA tracks, for which annotated corpora in form of question/answer pairs exists for multiple languages. These question types are also used in other recent data–driven QA approaches for evaluation, e.g., [10] or [11].

When–Answer Extraction In general, when–questions ask for instances of the EAT DATE. First, we replace the query terms with a star and remove all characters that are not numbers afterwards. We split the remaining string into substrings by means of star sequences. If the length of a substring is greater than three and if it contains a number, is added to the set of answer candidates. The value of the rank is given by $rank(\nu)$.

Who—Answer Extraction At the beginning, characters that are not letters are removed. Then, query terms and stop-words are replaced with a star. We split the remaining string into substrings by means of star sequences. If the substring contains at least one space and its frequency is greater than two, it is added to the set of answer candidates. Here, predicted answers which

contain "GEORGE BUSH" will be preferred to predicted answers which contain "BUSH", because they will have a higher correlation and therefore, a higher $rank(\nu)$.

Where–Answer Extraction is currently our most language–dependent part. This module uses geographical information about places around the world. Since we currently only make use of the English WordNet, we translate the NON–English answers (i.e., location names) using the Babelfish online MT service. The algorithm starts by removing all the characters that are not letters and we replace the query terms and stop-words with a star afterwards. We split the remaining string into substrings by means of star sequences. If the string is recognized by WordNet as a location, is added to the set of answer candidates. The value of the rank is given by $rank(\nu)$.

5 Experiments

We send a natural language Wh–question Q unmodified (i.e., without any pre–processing) to the Google search engine and extract the first 30 snippets. Each snippet is normalized by removing all HTML encoding, and by uppercasing the remaining text. We assessed the question/answering pairs from the multilingual CLEF 2004 corpus, which refers to answers from 1994/1995 newspaper articles. We consider two kinds of *correct answer*(CA):

Exact Answers(EA) are substrings that match one-to-one with the answers provided by CLEF. We should highlight that many CLEF answers are out of date and that often semantically valid alternative answers, i.e., those that are not expressed in the corpus, exist on the Web, often also decoded by using different spellings or word ordering.

Inexact Answer(IA) is an answer A that do not perfectly match with the answer A_c provided by CLEF, but for which there exists a close semantic relationship with A_c or where A corresponds to an update of A_c. For example, in case of WHERE–questions, which actually ask for a city name, we also accept the country name, and in case of WHO–questions, which requests the name of an official person, we accept the current one. Similarly, answers are also accepted, if they are just spelling variants, e.g., "George W. Bush", "G. Bush". In case of WHEN–questions, we also accept the answer "6 1945" or "1945", even though the exact answer in CLEF would be "6 August 1945".

We tested the system for 889 questions in four languages: English(EN), German(DE), Spanish(ES), and Portuguese (PT). The overall result for all languages can be inspected in Table 2. MRR stands for *Mean Reciprocal Rank*, and assigns to each question a score equal to the reciprocal of the rank of the first correct answer of the N (=3 in our case) best returned candidates. In the table, the results for the 1st, 2nd and 3rd place can be found, as well as for 0 (=NAF, which reads "no answer found, although there is one in the snippets"). Furthermore, NAG is when there was no answer in the snippets and

the system returns NIL, WAG is when there was no answer in the snippets and the system gave three wrong answers. Table 2 shows the results considering the four languages altogether and Table 3 the distribution of the extracted answers considering only when there was an answer in the snippets. Table 4 displays the results for the individual languages. For the German questions we only handled WHEN and WHERE questions, because for the WHO questions our simple "Wh–keyword spotting approach" does not work out due to Wh–keyword ambiguity.

Finally, a brief note on the performance of our system. The runtime for each question averaged over all the questions of the corpus is about 2881 milliseconds.

Table 2. Results for each question type over all languages

CA	Total	MRR	NAG(%)	WAG(%)	NAF(%)	1(%)	2(%)	3(%)
WHEN	218	0.60	25.11	10.96	21.46	35.16	5.02	1.8
WHERE	232	0.57	10.77	24.14	20.68	30.60	9.91	3.87
WHO	439	0.38	11.39	27.56	32.57	18.90	6.83	2.73

Table 3. Distribution of answer candidates (all languages)

CA	NAF(%)	1(%)	2(%)	3(%)
WHEN	33.82	55.42	7.91	2.84
WHERE	31.86	47.00	15.23	5.95
WHO	53.37	30.97	11.19	4.47

Table 4. The results for the individual languages

CA(EN)	Total	MRR	NAG(%)	WAG(%)	NAF(%)	1(%)	2(%)	3(%)
when	69	0.69	15.69	15.69	17.65	45.10	3.92	1.96
where	64	0.74	7.81	12.5	15.62	53.12	10.93	0
who	148	0.50	7.43	12.83	32.43	33.78	10.14	3.38
CA(DE)	Total	MRR	NAG(%)	WAG(%)	NAF(%)	1(%)	2(%)	3(%)
Wann	58	0.45	36.20	12.07	27.59	22.03	1.17	0
Wo	58	0.46	9.37	18.75	23.43	20.31	12.5	6.25
CA(ES)	Total	MRR	NAG(%)	WAG(%)	NAF(%)	1(%)	2(%)	3(%)
Cuándo	59	0.55	16.64	11.86	23.73	32.20	10.17	11.86
Dónde	63	0.59	10.93	31.25	15.62	26.56	10.93	3.21
Quién	86	0.27	9.65	40.68	28.96	11.72	6.21	2.75
CA(PT)	Total	MRR	NAG(%)	WAG(%)	NAF(%)	1(%)	2(%)	3(%)
Quando	56	0.04	30.76	12.30	42.45	3.08	1.54	0
Onde	47	0.18	10.93	25	20.31	10.93	1.56	4.68
Quem	146	0.14	17.12	29.45	36.30	10.95	4.11	2.05

For the individual question types, we obtained: 1) When-questions: 2505, 2) Where-questions: 5591, and 3) Who: 2613 milliseconds. The extra time for the Where–questions is caused by calling Babelfish.

6 Discussion

Due to the distribution shown in Table 3, most of the extracted answers were ranked at position one, and the very few external knowledge sources and linguistic tools used for our answer extraction module, we say that our current result for the predicted answers is encouraging. If we have a closer look to the results of the different question types, then our result is competitive with current alternative data–driven approaches of QA. For example, [10] present an instance–based approach to QA in which a system (for English, only) is automatically acquired using TREC data. In particular, for 296 TEMPORAL–questions from TREC 9–12 they obtain a MRR of 0.447 using a larger corpus than we and a stricter test (checking exact answers). Their result is consistently above the sixth highest score at each TREC 9–12. That leads us to claim that our predicted answers has at least a competitive quality.

Our result also suggest, that the answer prediction strategy does not behave similar for the different question types, and for different languages. We suspect that this is due to the very shallow nature of our current answer extraction modules, and because the distribution and redundancy of web pages per languages is very different. This is an important fact, because our ranking schema assigns sequences of highly frequent word pairs a larger eigenvalue and hence a stronger weight than sequences of less frequent word pairs, cf. Sect. 3.2. This means that sequences of highly frequent words will bias in a stronger way the length of the eigenvectors in the new orthogonal spaces.

Lets consider the following ratio:

$$\bar{G}(v) = \frac{G(v)}{\sum_{v=0}^{\omega} G(v)}$$

$\bar{G}(v)$ is the probability of pairs of words with a certain distance that occur v times in a web page. The following table shows some empirical values for $\bar{G}(v)$:

Frequency	$\bar{G}(v)$
0	0.999925
1	0.000685
2	0.00005
3 and more	0.000015

Stronger relations will occur much fewer than weaker relations, and thus express more about the content of a document. It also has the advantage for representing D with a small set of pairs of words.

7 Related Work

Current webQA systems mainly use statistical methods for finding answers from the web that exploit data redundancy rather than sophisticated linguistic analyses of either questions and candidate answers, cf. also [4]. [13] present an compact overview of current state–of–the art in webQA. They also present a query reformulation process designed for the Spanish language that uses a combination of simple string rewriting, following a generate–and–test approach, i.e., no answer source feedback is used. Although they used a small non–standard question/answer corpus for evaluation (40 factoid questions), the results look promising (MRR = 0.7175). A similar strategy was earlier investigated by [6] for answer extraction from English Web snippets obtaining a MRR=0.450 for 500 TREC–9 questions. [4] describe a feedback loop approach similar to ours, in which candidate answer terms were merged back into the query used for passage retrieval. The major difference compared to our approach is that they apply the feedback strategy after answer candidates have been determined, whereas we do it before answer extraction. They seem to perform the feedback loop on retrieved passages from TREC data only, which are less noisy in general than the snippets returned by Google. Furthermore, by not considering Web snippets, they can only make use of a reduced amount of redundancy, which might explain, why their approach was of less benefit as they expected. [12] present an approach for automatic derivation of surface text patterns using Maximum Entropy Modeling. They achieve a MRR=0.2993 on 500 TREC–10 questions. [11] presents a multilingual approach to QA using supervised Machine Learning algorithms (similar in spirit to [10], cf. Sect. 6). The methods extract answers as terms biased by the question using probabilistic models constructed from question–answer pairs. The results are promising (MRR=0.36 on 2000 Japanese question–answer pairs) Although all of the mentioned approaches consider only a single language, they support our perspective that language–independent statistical methods are essential for the development of multilingual QA system.

8 Conclusion

We presented a language independent strategy for predicting and extracting answers from Web snippets. We described a strategy that uses eigenvalues determined from a specialized designed matrix, which are used for determining the implicit semantic relationship between query and answer terms from the retrieved snippets. The matrix explicitly represents word–pairs and their distance. We evaluated our approach with three different types of questions from four languages, obtaining a combined MRR=0.52 for the respective subset of the CLEF–2004 data set. Currently, we are processing only simply Wh–questions. In future work we will perform more experiments taking into account additional types of questions and languages.

References

1. Moldovan, D., Harabagui, S., Clark, C., Bowden, M., Lehmann, J., Williams, J.: Experiments and Analysis of LCC's two QA Systems over TREC 2004. TREC 2004 (2004)
2. Radev, D.: Panel on web-based question answering. AAAI Spring Symposium on New Directions in Question Answering. (2003)
3. Neumann, G., Xu, F.: Mining natural language answers from the web. Web Intelligence and Agent Systems, volume **2**. (2004) 123–135
4. Clarke, C. L. A., Cormack, G. V., Lynam, T. R.: Exploiting Redundancy in Question Answering. SIGIR. (2001) 358-365
5. Ramakrishnan, G., Paranjape, D., Chakrabarti, S., Bhattacharyya, P.: Is Question Answering an Acquired Skill?. Proceedings of the 13th international conference on World Wide Web, WWW 2004. (2004)
6. Dumais, S. T., Banko, M., Brill, E., Lin, J. J., Ng, A. Y.: Web question answering: is more always better?. SIGIR. (2002) 291-298
7. Ravichandran, D., Hovy, E. H.: Learning surface text patterns for a Question Answering System. ACL (2002) 41-47
8. Belew, R. K.:Finding out About: A Cognitive Perspective on Search Engine Technology and the WWW. Cambridge University Press. (2000)
9. Robertson, S.:Understanding Inverse Document Frequency: On theoretical arguments for IDF. Journal of Documentation, volume **60**, number **5** (2004)
10. Lita, L., Carbonell, J.: Instance-Based Question Answering: A Data-Driven Approach. EMNLP 2004 (2004)
11. Sasaki, Y., Carbonell, J.: Question Answering As Question-Biased Term Extraction: a New Approach Toward Multilingual (QA). Proceedings of ACL (2005)
12. Ravichandran, D., Ittycheriah, A., Roukos, S.: Automatic Derivation of Surface Text Patterns for a Maximum Entropy Based Question Answering System. HLT-NAACL (2003)
13. Del-Castillo-Escobedo, A., Gómez, M., Villaseñor-Pineda, L.: QA on the Web: A Preliminary Study for Spanish Language. ENC-04 (2004)
14. Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., Harshman, R. A.: Indexing by Latent Semantic Analysis. Journal of the American Society of Information Science, volume **41**, number **6** (1990) 391-407