Towards Automatic Evaluation of Question/Answering Systems

**Bernardo Magnini, Matteo Negri, Roberto Prevete,
and Hristo Tanev**

ITC-irst, Centro per la Ricerca Scientifica e Tecnologica,
Via Sommarive, 3850 POVO (TN), Italy
{magnini,negri,prevete,tanev}@itc.it

Abstract

This paper presents an innovative approach to the automatic evaluation of Question Answering systems. The methodology relies on the use of the Web, considered as an "oracle" containing all the information needed to check the relevance of a candidate answer with respect to a given question. The procedure is completely automatic (i.e. no human intervention is required) and it is based on the assumption that the answers' relevance can be assessed from a purely quantitative perspective. The methodology is based on a Web search using patterns derived both from the question and from the answer. Different kinds of patterns have been identified, ranging from "lenient" (i.e. boolean combinations of single words), to "strict" patterns (i.e. whole sentences or combinations of phrases). A statistically-based algorithm has been developed which considers both the kinds of patterns used in the search and the number of documents returned from the Web. Experiments carried out on the TREC-10 corpus show that the approach achieves a high level of performance (*i.e.* 80% success rate).

1. Introduction

Textual Question Answering (QA) aims at identifying the answer to a question either on the Web or in a local document collection. QA systems are presented with natural language questions and the expected output is either the actual answer identified in a text or small text fragments containing the answer.

Automatic or semi-automatic evaluation techniques are of great interest for QA for several reasons. On one hand, most QA systems rely on complex architectures including modules in charge of the linguistic processing of the question, search in textual databases, and extraction of relevant text portions. Performance evaluation of such systems requires a huge amount of work. As a consequence, from the software engineering point of view, the automatic assessment of an answer with respect to a given question is important in the phases of algorithm refinement and testing. On the other hand, the availability of a completely automatic evaluation procedure makes feasible QA systems which are based on generate and test approaches. In this way, the system will carry out different refinements of its search criteria checking the relevance of new candidate answers until a given answer is automatically proved to be correct for a question.

Although there is some recent work addressing the evaluation of QA systems, it seems that the idea of using an automatic approach has still not been fully explored.

(Breck et al., 2000) presents a semi-automatic approach which relies on computing the overlap between the system response to a question and the stemmed content words of an answer key. Answer keys are manually constructed by human annotators using the TREC question corpus and external resources like the Web.

Several systems apply automatic answer validation techniques with the goal of filtering out improper candidates by checking how adequate an answer is with respect to a given question. These approaches rely on discovering semantic relations between the question and the answer. As an example, (Harabagiu and Maiorano, 1999) describes answer validation as an abductive inference process, where an answer is valid with respect to a question if a logical justification of its correctness can be provided. The justification is based on a backchaining proof from the logical form of the answer to the logical form of the question. Although theoretically well motivated, the use of semantic techniques on open domain tasks is quite expensive both in terms of the involved linguistic resources and in terms of computational complexity, thus motivating research on alternative solutions to the problem.

The approach we present attacks the problem from a new point of view, exploiting the amount of information present in the Web for a fully automatic answer validation process. Our methodology considers both the search criteria used for querying the Web and the number of hits. The underlying intuition is that, given a question/answer pair, the number of Web-retrieved documents in which the question and the answer keywords co-occur can be considered a significant clue to the validity of the answer. Following this intuition, different query formulation criteria ranging from *strict* (i.e. close to the actual form of a question) to more *lenient* (i.e. traditional word level queries) search patterns are possible. The trade-off between the use of different search patterns and the number of documents retrieved is thoroughly investigated in order to maximize the effectiveness of a statistical approach.

The paper is structured as follows. Section 2 introduces the problem of the evaluation of QA systems, fixing the context of this work. Section 3 illustrates the concept of *validation patterns* and presents the basic distinction between *lenient* and *strict* patterns. Section 4 describes a simple pattern extraction technique. Section 5 addresses the problem of statistical answer validation, presenting an algorithm for estimating an answer's relevance. Section 6 discusses the results of an automatic evaluation experiment carried out using the TREC-10 question corpus as data set.

2. Answer Relevance

The evaluation of QA systems can be carried out from different perspectives, taking into consideration either their usability and the time of execution or the "quality" of the output they provide. Until now, most of the effort in developing QA systems still deals with the second aspect of the challenge, focusing on the problem of an answer's "correctness". Although the QA roadmap (Burger et al., 2001) includes the analysis of other aspects of the task (real-time QA, interactive QA, user profiling, etc.) in the future, so far the TREC competition guidelines only asked participants to provide "correct" answers to open-domain natural language questions.

The definition of correctness depends both on the kind of questions systems have to deal with, and on the complexity of the required answers. Some questions are much harder than others because they require explanations (*e.g.* "How to assemble a computer?") or definitions (*e.g.* "Who was Galileo?"). In these cases it is not clear what makes a good answer because of the inherent ambiguity of the questions. (Harabagiu et al., 2000) points out that for the same question, the answer may be easier or more difficult to extract depending on how it is phrased in the text. For instance lists (*e.g.* "Name 9 countries that import Cuban sugar"), opinions (*e.g.* "Should the Fed raise interest rates at their next meeting?") and summaries (*e.g.* "What are the arguments for and against prayer in school?") are difficult to provide because they require dealing with information scattered throughout a document or across multiple documents. In these cases, the problem of finding answers depends on factors (for example reasoning mechanisms and knowledge of the world) that exceed traditional natural language processing techniques. As a consequence, the definition of an answer's correctness is also quite problematic because it is not a simple text portion found verbatim in sentences or paragraphs.

So far, the difficulty of dealing with such kinds of questions and answers has confined most of the research efforts to the study of questions with short factual answers (*e.g.* "When did Elvis Presley die?"). Different criteria have been proposed for judging this kind of answers. For instance, (Hirschman and Gaizauskas, 2001) considers relevance, correctness, conciseness, completeness, coherence and justification as possible features of a good answer and points out that, according to the TREC evaluation criteria, so far evaluations have focused primarily on relevance (i.e. is the answer actually responsive to the question?) and partially on justification (i.e. is the answer supplied with sufficient context to allow a reader to determine why this was chosen as an answer to the question?).

This work is focused on the first of these two aspects: given a question q and a candidate answer a, we define the answer validation task as the capability to assess the relevance of a with respect to q. We assume fact-based open-domain questions and that both answers and questions are texts composed of few tokens (usually less than 100). This is compatible with the TREC-10 data, which will be used as an example throughout this paper. We also assume the availability of the Web, considered to be the largest open domain text corpus containing information about almost all areas of the human knowledge.

3. Validation Patterns

The use of patterns has recently emerged as an interesting approach to the QA task. For example (Soubbotin and Soubbotin, 2001) describes a QA system based on searching for predefined patterns of textual expressions that may be interpreted as answers to certain types of questions. Given a question and a list of candidate answers, the system considers as correct only the text portions matching any of the pattern strings corresponding to the question type category.

Our approach to answer validation benefits greatly from using patterns. The motivation for a pattern-based approach is that patterns represent an effective way for capturing both explicit and implicit information from text documents. Answers, in fact, may occur in text passages with low similarity with respect to the question. Passages containing facts' descriptions may use different syntactic constructions, sometimes are spread in more than one sentence, may reflect opinions and personal attitudes, and often use ellipsis and anaphora. For instance, consider the question "What is the capital of the USA?" and the answer "Washington". Using patterns we can find Web documents containing passages like those reported in Table 1, which contain a significant amount of knowledge about the relations between the question and the answer. We will refer to these text fragments as *validation fragments*.

1. Capital Region USA: Fly-Drive Holidays in and Around Washington D.C.
2. the Insider's Guide to the Capital Area Music Scene (Washington D.C., USA).
3. The Capital Tangueros (Washington, DC Area, USA)
4. I live in the Nation's Capital, Washington Metropolitan Area (USA).
5. in 1790 Capital (also USA's capital): Washington D.C. Area: 179 square km

Table 1: Validation fragments

Given a question/answer pair, we use the term *validation patterns* to indicate queries built with different combinations of words extracted both from the question and from the answer. The underlying intuition is that if a certain answer is relevant with respect to a given question, then a Web search using validation patterns will result in a large set of documents connecting question and answer concepts. On the contrary, validation patterns built combining the question keywords with keywords extracted from a wrong answer will lead to few documents.

Considering the above example, the problem of validating the answer "Washington" can be tackled searching the Web for documents in which the concepts "capital of the USA?" and "Washington" co-occur. Our hypothesis is that the co-occurrence tendency of question and answer concepts (i.e. the number of retrieved documents) can be considered a significant clue to the validity of the answer.

Different kinds of patterns can be used in order to cover a large portion of validation fragments, including those lexically similar to the question and the answer (*e.g.* fragments 4 and 5 in Table 1) and also those that are not similar (*e.g.* fragment 2 in Table 1). A search query covering all the validation fragments of our example can be generalized by the pattern:

[capital <text> USA <text> Washington]

where <text> is a place holder for any portion of text.

The construction of a validation pattern is not straightforward. The advanced search modalities of modern search engines allow the user to put together query keywords in a number of ways, narrowing or enlarging the search space. Validation patterns can be classified according to their degree of generality, and range from *lenient* (i.e. boolean compositions of single words) to *strict* (i.e. search queries using strings) patterns.

3.1. Lenient Patterns

Lenient patterns, or word level patterns, represent the easiest way of combining the search keywords. At this level question and answer keywords are simply put together as separate words using boolean (i.e. AND/OR operators) or more sophisticated operators (*e.g.* the NEAR operator provided by AltaVista, which searches for pages where two words appear in a distance of no more than 10 tokens).

For example, consider the TREC-10 question: "Who discovered X rays?" and the correct answer "Roentgen". The following lenient patterns, in decreasing order of generality, can be created combining question and answer keywords:

[discovered OR X OR rays OR Roentgen]
[discovered AND X AND rays AND Roentgen]
[discovered NEAR X NEAR rays NEAR Roentgen]

The first two patterns are the most general, and retrieve respectively pages containing at least one of the keywords and pages containing all the search keywords. The third pattern represents a further narrowing of the search space, and leads to documents containing all the query keywords, each of which separated by at most 10 tokens.

A common feature of lenient patterns is that they usually produce a large number of hits but present a low degree of reliability. Even in the case of retrieved documents containing all the search keywords, there is no guarantee that the relation between these concepts is of the expected type. As an example, lenient patterns built with the AltaVista NEAR operator lead respectively to 414 documents justifying the right answer (i.e. "Roentgen") and five documents for the wrong answer "Einstein". One of the phrases covered by the wrong answer pattern is:

"Historical Timeline. 1895 - Wilhelm Roentgen discovered X-rays. 1905 - Albert Einstein developed the theory about the relationship of mass and energy".

The fact that "Einstein" is in proximity of "discovered", "X" and " rays" is enough for the word level pattern to cover the text even if the co-occurrence of the these words is purely accidental.

3.2. Strict patterns

Strict patterns represent a refinement of the searching criteria which combines question and answer keywords ranging from the *phrase level* to the *sentence level*. As a result, the casual co-occurrence of query keywords is considerably reduced, while co-occurrences resulting from a semantic relation between these concepts will tend to remain stable. For instance, considering the verb-phrase "discovered X rays" as a search unit, our question about X-rays can be combined with the respective correct answer into the following strict patterns:

["discovered X rays" OR Roentgen]
["discovered X rays" AND Roentgen]
["discovered X rays" NEAR Roentgen]

Since the string "discovered X rays" is less likely to appear in a written text with respect to any of the unordered combinations of the words "discover", "X" and "rays", we would expect that strict patterns will cover a smaller portion of Web documents. However, we expect that the right answer "Roentgen" will continue to appear in the context of the verb-phrase, while the wrong answer "Einstein" would disappear or rarely appear in the same context. A new Web search using the NEAR operator provided by AltaVista confirms our hypothesis. In fact, the verb-phrase "discovered X rays" appears close to the answer "Roentgen" in 215 documents, while it appears close to "Einstein" only in one document.

Further refinements of the search criteria are possible in order to create more and more strict patterns. For example, the question analysis of "Who discovered X rays?" shows that we are looking for a person, which may be the subject of a sentence whose verb phrase is "discovered X rays". Using syntactic information we can move from the phrase level to the sentence level in order to build the following stricter patterns:

["Roentgen discovered X rays"]
["Einstein discovered X rays"]

A Web search using the first pattern returns 106 documents, while the second pattern is not found on the Web.

The trade-off between the use of different search patterns and the number of documents retrieved has to be considered for statistically-based answer validation. The longer the phrases or sentences that the patterns contain, the less probable they become and the risk of missing a right answer increases. Moreover, the small number of text fragments covered by strict patterns makes the application of any statistical approach difficult. On the other hand, the less probable the pattern is, the higher the information value of its appearance in a text collection. Therefore, the appearance of a low-probability pattern can be considered a significant clue to the validity of an answer. The different ways in which facts can be reformulated makes the very strict patterns inapplicable in some cases.

4. Using Patterns for Answer Validation

In the experiments reported in this paper we have used only one type of the patterns mentioned above, namely phrase-level patterns. This kind of pattern represents a good test for checking the validity of the statistical answer validation technique. Moreover, phrase-level patterns allow for a good compromise between the number of documents returned from the Web and the possibility of casual co-occurrence of the query keywords.

From this point on, we will use *Qsp* to denote the question sub-pattern (i.e. the portion of pattern derived from the input question) and *Asp* to denote the answer sub-pattern (i.e. the portion of the pattern derived from a candidate answer).

Question sub-pattern (*Qsp*). During the construction of the *Qsp* phrases are identified by means of a shallow parser. Keywords are then expanded with both synonyms and morphological forms in order to maximize the recall of retrieved documents. Synonyms are automatically extracted from the most frequent sense of the word in WordNet (Fellbaum, 1998), which considerably reduces the risk of adding disturbing elements. As for morphology, verbs are expanded with all their tense forms (*i.e.* present, present continuous, past tense and past participle). Synonyms and morphological forms are added to the *Qsp* and combined in an OR clause. As an example, consider the following TREC-10 question: "George Bush purchased a small interest in which baseball team?" There are two uninterrupted sequences to be considered here: "George Bush purchased a small interest" and "baseball team". After the keyword expansions phase described above, the two token sequences are combined in the following *Qsp*:
[George Bush (purchased OR purchase OR purchasing OR buy OR bought...) (small OR little) (interest OR involvement) <text> (baseball OR ball-game) (team OR squad)]

Answer sub-pattern (*Asp*). An *Asp* is constructed in two steps. First, the *answer type* of the question is identified considering both morpho-syntactic (a part of speech tagger is used to process the question) and semantic features (by means of semantic predicates defined on the WordNet taxonomy; see (Magnini et al., 2001) for details). Possible answer types are: DATE, MEASURE, NAME, DEFINITION and GENERIC. The first two categories are obviously related to questions asking respectively for dates and measures. NAME is a broad class related to questions asking for person, organization or location names. DEFINITION is the answer type peculiar to questions like "What is an atom?" which represent a considerable part (around 25%) of the TREC-10 corpus. The answer type GENERIC is used for non-definition questions asking for entities that can not be classified as names, dates or measures (*e.g.* the questions: "Material called linen is made from what plant?" or "What mineral helps prevent osteoporosis?")

In the second step, a rule-based named entities recognition module identifies in the answer string all the named entities matching the answer type category. If the category is NAME, DATE or MEASURE, this step of the algorithm creates an *Asp* for every selected named entity. If the answer type category is DEFINITION or GENERIC, the entire answer string except the stop-words is considered. In addition, in order to maximize the recall of retrieved documents, the *Asp* is expanded with verb tenses. The following example shows how the *Asp* is created. Given the TREC question "When did Elvis Presley die?" and the candidate answer "though died in 1977 of course some fans maintain", since the answer type category is DATE the named entities recognition module will select [1977] as an answer sub-pattern.

5. Estimating Answer Validity

In contrast with qualitative approaches to Web mining (*e.g.* (Brill et al., 2001)) based on the analysis of the retrieved documents' content, we use a quantitative algorithm that considers the number of hits. As a result of this approach, we consider a large number of pages whereas qualitative approaches can only work on a relatively small number.

5.1. Querying the Web

The answer validation module submits three searches to the search engine: the sub-patterns [*Qsp*] and [*Asp*] and the validation pattern [*QAp*] built as the composition of the two sub-patterns using the AltaVista NEAR operator. Afterwards, a statistical algorithm considers the output of the Web search for estimating the consistency of the patterns.

Several pattern relaxation heuristics have been defined in order to gradually increase the number of retrieved documents. If the question sub-pattern Qsp does not return any document or returns less than a certain threshold (experimentally set to 7) the question pattern is relaxed by cutting one word; in this way a new query is formulated and submitted to the search engine. This is repeated until no more words can be cut or the returned number of documents becomes higher than the threshold.

Pattern relaxation is performed using word-ignoring rules in a specified order. Such rules, for instance, ignore the question focus, because it is unlikely that it occurs in a validation fragment; ignore adverbs and adjectives, because they are less significant; and ignore nouns belonging to the WordNet classes "abstraction", "psychological feature" or "group", because usually they specify finer details and human attitudes. Names, numbers and measures are preferred over all the lower-case words and are cut last.

5.2. Estimating pattern consistency

As a result of the Web search with patterns, the search engine returns three sets of documents: $hits(Qsp)$, $hits(Asp)$ and $hits(Qsp$ NEAR $Asp)$. The probability $P(A)$ of a pattern A in the Web is calculated by:

$$P(A) = \frac{hits(A)}{NWeb-pages}$$

where $hits(A)$ is the number of pages in the Web where A appears and $NWeb-pages$ is the maximum number of pages that can be returned by the search engine. We set this constant experimentally. However in the both formulas we use (*i.e.* Pointwise Mutual Information and Corrected Conditional Probability) $NWeb-pages$ may be ignored.

The joint probability P(*Qsp*,*Asp*) is calculated by means of the validation pattern probability:

$$P(QAp) = P(Qsp\texttt{NEAR}Asp)$$

We have tested two alternative measures to estimate the degree of relevance of Web searches: Pointwise Mutual Information and Corrected Conditional Probability, a variant of Conditional Probability which considers the asymmetry of the question-answer relation. Each measure provides an answer validity score: high values are interpreted as strong evidence that the validation pattern is consistent. This is a clue to the fact that Web pages where this pattern appears contain validation fragments which support the candidate answer.

Pointwise Mutual Information (PMI). PMI (Manning and Schütze, 1999) has been widely used to find co-occurrence in large corpora.

$$PMI(\textit{Qsp,Asp}) = \frac{P(\textit{Qsp,Asp})}{P(\textit{Qsp}) * P(\textit{Asp})}$$

PMI(Qsp,Asp) is used as a clue to the internal coherence of the question-answer validation pattern *QAp*. Substituting the probabilities in the PMI formula with the previously introduced Web statistics, we obtain:

$$\frac{hits(\textit{Qsp}\ \texttt{NEAR}\ \textit{Asp})}{hits(\textit{Qsp}) * hits(\textit{Asp})} * NWeb\text{–}pages$$

Corrected Conditional Probability (CCP). In contrast with PMI, CCP is not symmetric (*e.g.* generally $CCP(Qsp, Asp) \neq CCP(Asp, Qsp)$). This is based on the fact that we search for the occurrence of the answer pattern Asp only in the cases when *Qsp* is present. The statistical evidence for this can be measured through $P(Asp|Qsp)$, however this value is corrected with $P(Asp)^{2/3}$ in the denominator, to avoid the cases when high-frequency words and patterns are taken as relevant answers.

$$CCP(Qsp, Asp) = \frac{P(Asp|Qsp)}{P(Asp)^{2/3}}$$

For CCP we obtain:

$$\frac{hits(Qsp\ \texttt{NEAR}\ Asp)}{hits(Qsp) * hits(Asp)^{2/3}} * NWeb\text{–}pages^{2/3}$$

5.3. An Example

Consider the TREC-10 question: “Which river in US is known as Big Muddy?”. The search of the question sub-pattern:

[river NEAR US NEAR (known OR know) OR NEAR “Big Muddy”]

returns 0 pages, so the algorithm relaxes the pattern by cutting the initial noun “river”, according to the heuristic for discarding a noun if it is the first keyword in the question. The second *Qsp* also returns 0 pages, so we apply the heuristic for ignoring verbs like “know”, “call” and abstract nouns like “name”. The third *Qsp* [US NEAR “Big Muddy”] returns 28 pages, which is over the experimentally set threshold of seven pages.

One of the possible TREC-10 candidate answers is “Mississippi River”. To calculate answer validity score (in this example PMI) for [“Mississippi River”], the procedure constructs the validation pattern:

[US NEAR “Big Muddy” NEAR “Mississippi River”]

as a conjunction of *Qsp* and the answer sub-pattern [“Mississippi River”]. These two patterns are passed to the search engine, and the returned numbers of pages are substituted in the mutual information expression at the places of $hits(Qsp, \texttt{NEAR}, Asp)$ and $hits(Asp)$ respectively; the previously obtained number (*i.e.* 28) is substituted at the place of $hits(Qsp)$. In this way an answer validity score of 55.5 is calculated.

6. Experiments and Discussion

A number of experiments have been carried out in order to check the correctness of the proposed answer validation technique. As a data set, the 492 questions of the TREC-10 database have been used. For each question, at most three correct answers and three wrong answers have been randomly selected from the TREC-10 participants’ submissions, resulting in a corpus of 2726 question/answer pairs (some question have less than three positive answers in the corpus). As mentioned previously, AltaVista was used as the search engine.

A baseline for the answer validation experiment was defined by considering how often an answer occurs in the top 10 documents among those (1000 for each question) provided by NIST to TREC-10 participants. An answer was judged correct for a question if it appears at least one time in the first 10 documents retrieved for that question, otherwise it was judged wrong. Baseline results are reported in Table 2.

Three independent factors were considered in the experiment:

Estimation method. We have implemented two measures (reported in Section 5.2.) to estimate an answer validity score: PMI and CCP.

Threshold. We wanted to estimate the role of two different kinds of thresholds for the assessment of answer validation. In the case of an *absolute threshold*, if the answer validity score for a candidate answer is below the threshold, the answer is considered wrong, otherwise it is accepted as relevant. In a second type of experiment, for every question and its corresponding answers the program chooses the answer with the highest validity score and calculates a *relative threshold* on that basis (*i.e.* $threshold = k * Max\_Validity\_score$). However the relative threshold should be larger than a certain minimum value.

Question type. We wanted to check performance variation based on different types of TREC-10 questions. In particular, we have separated definition and generic questions from factual questions whose actual answer is a named entity.

Tables 2 and 3 report the results of the automatic answer validation experiments obtained respectively on all the TREC-10 question corpus and on the subset of questions asking for named entities. For each estimation method we report precision, recall and success rate. Success rate best represents the performance of the system, being the percent of $[q, a]$ pairs where the result given by the system is the same as the TREC judges' opinion. Precision is the percent of $[q, a]$ pairs estimated by the algorithm as relevant, for which the opinion of TREC judges was the same. Recall shows the percent of the relevant answers which the system also evaluates as relevant.

	P (%)	*R (%)*	*SR (%)*
Baseline	50.86	4.49	52.99
PMI - abs.	70.65	84.36	76.76
PMI - rel.	79.27	73.61	79.08
CCP - abs.	76.96	75.30	78.35
CCP - rel.	79.36	77.71	80.52

Table 2: Results: all 492 TREC-10 questions

	P (%)	*R (%)*	*SR (%)*
PMI - abs.	80.13	79.77	82.01
PMI - rel.	86.40	73.67	83.05
CCP - abs.	85.19	73.34	82.38
CCP - rel.	87.71	76.82	84.83

Table 3: Results: 249 named entity questions

The best results on the 492 questions corpus (CCP measure with relative threshold) show a success rate of 80.5%, *i.e.* in 80.5% of the pairs the system evaluation corresponds to the human evaluation, and confirms the initial working hypotheses. This is 27.5% above the baseline success rate. Precision and recall are respectively 20-29% and 71-80% above the baseline values. These results demonstrate that the intuition behind the approach is well-motivated and that the algorithm provides a workable solution for answer validation.

The experiments show that the average difference between the success rates obtained for the named entity questions (Table 3) and the full TREC-10 question set (Table 2) is 4.4%. This means that our approach performs better when the answer entities are well specified.

Another conclusion is that the relative threshold demonstrates superiority over the absolute threshold in both test sets (average difference 2%). However if the percent of the right answers in the answer set is lower, then the efficiency of this approach may decrease.

The best results in both question sets are obtained by applying CCP with a relative threshold. Such non-symmetric formulas might turn out to be more applicable in general. As Corrected Conditional Probability (CCP) is not a classical co-occurrence measure like PMI, we may consider its high performance as proof of the difference between our task and classic co-occurrence mining. It seems that other measures are necessary for question-answer co-occurrence mining.

7. Related Work

The idea of using the Web as a corpus is an emerging topic of interest among the computational linguistics community. The TREC-10 QA track demonstrated that Web redundancy can be exploited at different levels in the process of finding answers to natural language questions. Several studies (*e.g.* (Clarke et al., 2001) (Brill et al., 2001)) suggest that the application of Web searching can improve the precision of a QA system by 25-30%. A common feature of these approaches is the use of the Web to introduce data redundancy for a more reliable answer extraction from local text collections. (Radev et al., 2001) suggests a probabilistic algorithm that learns the best query paraphrase of a question searching the Web. Other approaches suggest training a question-answering system on the Web (Mann, 2001).

The Web-mining algorithm presented in this paper is similar to PMI-IR (Pointwise Mutual Information - Information Retrieval) as described in (Turney, 2001). Turney uses PMI and Web retrieval to decide which word in a list of candidates is the best synonym with respect to a target word. However, the answer validity task poses different peculiarities. We search how the occurrence of the question words influence the appearance of answer words. Therefore, we introduce additional linguistic techniques for pattern and query formulation, such as keyword extraction, answer type extraction, named entities recognition and pattern relaxation.

8. Conclusion and Future Work

We have presented an approach to answer validation based on the intuition that the amount of implicit knowledge which connects an answer to a question can be quantitatively estimated by exploiting the redundancy of Web information. Results obtained on the TREC-10 QA corpus correlate well with the human assessment of answers' correctness and confirm that a statistical Web-based algorithm provides a workable solution for answer validation.

Several activities are planned in the near future. First, a deeper analysis of the trade-off between the use of different levels of patterns and the number of documents retrieved from the web will be carried out. We plan to test algorithms capable of deciding about the relevance of an answer considering both the high value of a small number of hits provided by a very strict pattern and the implicit information conveyed by the high number of hits provided by word-level patterns.

Second, a generate and test module based on the validation algorithm presented in this paper will be integrated into the architecture of our QA system under development. In order to exploit the efficiency and the reliability of the algorithm, such a system will be designed to maximize the recall of retrieved candidate answers. Instead of performing a deep linguistic analysis of these passages, the system will delegate to the evaluation component the selection of the right answer.

9. References

E. J. Breck, J.D. Burger, L. Ferro, L. Hirschman, D. House, M. Light, and I. Mani. 2000. How to Evaluate Your

Question Answering System Every Day and Still Get Real Work Done. In *Proceedings of LREC-2000*, pages 1495–1500, Athens, Greece, 31 May - 2 June.

E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. 2001. Data-Intensive Question Answering. In *TREC-10 Notebook Papers*, Gaithesburg, MD.

J. Burger, C. Cardie, V. Chaudhri, R. Gaizauskas, S. Harabagiu, D. Israel, C. Jacquemin, C.-Y. Lin, S. Maiorano, G. Miller, D. Moldovan, B. Ogden, J. Prager, E. Riloff, A. Singhal, R. Shrihari, T. Strzalkowski, Voorhees E, and R. Weishedel. 2001. Issues, Tasks and Program Structures to Roadmap Research in Question Answering (QA). *Available at: http://www-nlpir.nist.gov/projects/duc/roadmapping.html*.

C. Clarke, G. Cormack, T. Lynam, C. Li, and G. McLearn. 2001. Web Reinforced Question Answering (MultiText Experiments for TREC 2001). In *TREC-10 Notebook Papers*, Gaithesburg, MD.

C. Fellbaum. 1998. *WordNet, An Electronic Lexical Database*. The MIT Press.

S. Harabagiu and S. Maiorano. 1999. Finding Answers in Large Collections of Texts: Paragraph Indexing + Abductive Inference. In *Proceedings of the AAAI Fall Symposium on Question Answering Systems*, pages 63–71, November.

S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Grju, V. Rus, and P. Morarescu. 2000. FALCON: Boosting Knowledge for Answer Engines. In *Proceedings of the TREC-9 Conference*, pages 479–487.

L. Hirschman and R. Gaizauskas. 2001. Natural Language Question Answering: the View from Here. *Natural Language Engineering*, 7(4):275–300, December.

B. Magnini, M. Negri, R. Prevete, and H. Tanev. 2001. Multilingual Question/Answering: the DIOGENE System. In *TREC-10 Notebook Papers*, Gaithesburg, MD.

G. S. Mann. 2001. A Statistical Method for Short Answer Extraction. In *Proceedings of the ACL-2001 Workshop on Open-Domain Question Answering*, Toulouse, France, July.

C. D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT PRESS, Cambridge, Massachusets.

D. R. Radev, H. Qi, Z. Zheng, S. Blair-Goldensohn, Z. Zhang, W. Fan, and J. Prager. 2001. Mining the Web for Answers to Natural Language Questions. In *Proceedings of 2001 ACM CIKM*, Atlanta, Georgia, USA, November.

M. M. Soubbotin and S. M. Soubbotin. 2001. Patterns of Potential Answer Expressions as Clues to the Right Answers. In *TREC-10 Notebook Papers*, Gaithesburg, MD.

P. D. Turney. 2001. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of ECML2001*, pages 491–502, Freiburg, Germany.