Self-Learning Web Question Answering System

Dmitri Roussinov
Arizona State University
P.O Box 873606
Tempe, AZ, 85287
(480)965-8488
dmitri.roussinov@asu.edu

Jose Robles
Arizona State University
P.O Box 873606
Tempe, AZ, 85287
(480)965-3252
Jose.Robles@asu.edu

ABSTRACT

While being quite successful in providing keyword based access to web pages, commercial search portals, such as Google, Yahoo, AltaVista, and AOL, still lack the ability to answer questions expressed in a natural language. In this paper, we present a probabilistic approach to automated question answering on the Web. Our approach is based on pattern matching and answer triangulation. By taking advantage of the redundancy inherent in the Web, each answer found by the system is triangulated (confirmed or disconfirmed) against other possible answers. Our approach is entirely self-learning: it does not involve any linguistic resources, nor it does require any manual tuning. Thus, the propose approach can easily be replicated in other information systems with large redundancy.

1. INTRODUCTION

The goal of question answering (QA) is to identify and present to the user an actual answer to a question formulated in a natural language (e.g. English), rather than identifying documents that may be topically related to the question or may contain the answer. In contrast to the earlier QA approaches that rely on laboriously created linguistic resources, "shallow" approaches that use only simple pattern matching and inherent redundancy of such large repositories as, for example World Wide Web, have been recently successfully tried. Dumais et. al [2] presented a Web based QA system that uses simple combinatorial permutations of words (so called "re-writes") and a set of 15 handcrafted semantic filters to achieve a striking accuracy: Mean Reciprocal Rank (MRR) of *0.507*, which can be roughly interpreted as "in average" the correct answer being the second answer found by the system. Our work expands [2] by allowing automatic identification and training a set of patterns, which are more powerful than simple re-writes. We also explore the idea of using similar pattern matching algorithms on the Web to validate semantic categories of the answers (e.g. for a question *"In which city Eiffel Tower is located?"* the semantic category is *city*.) Our approach relies only on simple machine learning techniques, rather than on manually crafted rules or expensive linguistics resources.

2. ALGORITHMS

The general idea behind the proposed approach is pattern matching. For example, an answer for the question *"Who is the CEO of IBM?"* can be found in a sentence *"The CEO of IBM is Samuel Palmisano."* which matches a pattern *\Q is \A* where \Q is a *question part* ("The CEO of IBM") and *\A* = "*Samuel Palmisano*" is the text that forms a *candidate answer*. We automatically create and train up to *200* patterns for each question type (such as *what is, what was, where is*, etc.) based on training data set consisting of Q/A pairs. Through training, each pattern is assigned the probability that the matching text contains the correct answer. This probability is used in the triangulation (confirming/disconfirming) process that re-ranks the candidate answers. *\A, \Q, \T, \p* (punctuation mark) *\V* and * (a wildcard that matches any words) are the only special symbols used in our pattern language so far. Our approach allows adding more symbols if needed.

2.1 Question Answering Steps

Answering the question *"In which city is Eiffel Tower located?"* demonstrates the steps of our algorithm, omitting some details due to the space limitations.

Type Identification. The question itself matches the pattern *In which \T is \Q \V*, where *\T* = *"city"* is the semantic category of the expected answer, *\Q* = *"Eiffel Tower"* is the question part, and *\V* is a past tense verb, used in some types of the questions. We use freely available trainable part of speech (POS) tagger [6] to interpret questions only. Its use is needed only for a small minority of question types. The web pages are not tagged while searching for the answers.

Query modulation converts each answer pattern (e.g. *\Q is \V in \A*) into a query for a general purpose search engine (GPSE). For example, the AltaVista query would be " *"Eiffel Tower is" NEAR located"*.

Answer Matching. The sentence *"Eiffel Tower is located in the center of Paris, the capital of France."* would result in a match and creating a candidate answer "the center of Paris, the capital of France" with a corresponding probability of containing a correct answer (e.g. *.5*) obtained previously for each pattern by training as discussed below.

Answer Detailing forms more candidate answers by taking subphrases from the initial candidate answers. Our subphrases do not exceed 3 words (not counting "stopwords" such as *a, the, in, on*) and do not cross punctuation marks. Each sub-phrase candidate answer is assigned the same score as the original candidate answer multiplied by the proportion of the length of the subphrase (measured in words) relatively to the original match. In our example that would be "center" (*.125*), "Paris" (*.125*), "capital" (*.125*), "France" (*.125*), "center of Paris" (*.5*), "capital of France" (*.5*).

The algorithm stops querying GPSE when a specified number of web pages has been scanned (1000 in this study). If there fewer than expected (here 200) candidate answers found, the algorithm resorts to a "fall back" approach as in [2]: it creates candidate answers from each sentence of the snippets returned by GPSE and applies answer detailing to them. If there are still not enough candidates, the system automatically relaxes the modulated query (by removing most frequent on the Web words first) until enough many hits are returned by GPSE. This way, it can simplify a question *"Who still makes rod hockey games?"* to *"Who still makes rod hockey?"*

Triangulation. The candidate answers are triangulated (confirmed or disconfirmed) against each other, as detailed in the next section, then re-ordered according to their final score.

Semantic Filtering. The system performs a search on the Web for confirming the semantic type of the top 20 candidates, by following the same steps as if it were answering the question *"What is Paris?"* and expecting an answer *a city*. For example, the sentence

"Paris, a city of dreams, can be amazing at night." matches the following WHAT-IS answer pattern: \Q , \A *. Several matches may be required to establish that "Paris" is indeed a *city* until the desired certainly is reached. The task of confirmation is easier than answering since the sought answer is already known ("*a city*"). We did not involve any manually crafted semantic filters reported in prior studies [2] such as those dealing with numbers, proper names, and monetary values.

2.2 Triangulation

Triangulation, a term widely used in intelligence and journalism, stands for confirming or disconfirming facts using multiple sources. We went one step further than using frequency counts explored earlier in [2] and tried a more fine-grained triangulation process. Our algorithm can be demonstrated by the following intuitive example. Imagine that we have two candidate answers for the question *"What was the purpose of Manhattan Project?"* 1) *"To develop a nuclear bomb"* and 2) *"To create a nuclear weapon."* Those two answers should reinforce (triangulate) each other since they are semantically similar, however a straightforward frequency count approach would not pick this similarity. The advantage of triangulation over simple frequency counting is stronger for less "factual" questions, those that may allow variation in the correct answers. This includes such frequent types as definitions and "how to" questions. Although, there are many known measures of semantic similarity between words and phrases, for simplicity, we used *relative significant overlap* in the current implementation:

sim (a1, a2) = so(a1, a2)/(length(a1)+length(a2)), where *so(a1,a2)* is the number of words that are present in both *a1* and *a2*, that are not stopwords and not the words from the question part. In the above example *sim = 1/(3+3)*. We are currently exploring more "semantic" measures, e.g. those produced from mutual co-occurrence statistics. The resulting score for each candidate answer $s^t(a)$ after triangulation is computed by summation: $s^t(a) = \sum_{a_i \in O,\, a_i \neq a} s(a_i) \cdot sim(a, a_i)$, where *O* is the set of all original (before detailing) answers and *s(a)* is the original score.

2.3 Scalability and Responsiveness

Since our objective was to explore the feasibility of the approach, we were not that much concerned with real-time responsiveness. Our proof of concept prototype finds an answer within minutes. The bottleneck is fetching the content of Web pages, which can be parallelized on multiple workstations that would send to the central server only the identified candidate answers, as for example has been successfully demonstrated in [4]. Another solution is to have direct access to GPSE index and cache, which may be, for example, feasible when QA system is an internal part of it.

2.4 Pattern Training

For each training Q/A pair, the system requests web pages from GPSE that have both the question (Q) and the answer (A), preferably in proximity. Each sentence containing both Q and A is converted into a candidate pattern by replacing Q-part (e.g. "Eiffel Tower") with \Q symbol and the answer ("Paris") with \A. Once a specified number of candidate patterns is identified (200 in our experiments), more patterns are generated through a recursive "generalization" process of replacing words with wildcards and forming substrings containing both \Q and \A. Sequences of adjacent wildcards are merged and all the wildcards next to \A are removed since redundant. The obtained top most frequent 500 patterns are trained for the probability of matching the text that includes a correct answer by the modulation and matching process similar to the described above. The fall back approach probabilities are also similarly trained. Unsupervised pattern training approach on the Web was first proposed by [7] but they did not use wildcards nor the fall back approach.

3. EMPIRICAL EVALUATION AND CONCLUSIONS

We used TREC Q/A data sets [5] from 1999 to 2002 for training, except the year 2001, which we used for testing in order to compare to the prior results. We achieved mean reciprocal rank of the answer (MRR) of *0.314*, which is smaller than reported in [2], but seemingly better (although not directly comparable due to different testing methodologies) to the results reported with the other approaches [3][1] that were completely trainable (not relying on hand-crafted rules). We believe that the lower results are due to our use of AltaVista instead of Google and due to the fact that we did not use any manually crafted semantic filters. Without them, [2] reported MRR of *0.416*. Also, our test set had only a few questions that may benefit from our automated semantic filter. It included mostly clearly defined answers, the scenario when our triangulation mechanism does not have much of an advantage over simple frequency count. We have not yet comprehensively tested each component separately. Nevertheless, we believe that our preliminary results are encouraging since our approach does not require any manual tuning, thus replicable by follow-up studies and easily implemented in other open domain knowledge management/search/retrieval systems. Our empirical pilot study with users searching the Web for the answers to the given questions is currently on the way. The preliminary feedback from several uses who tried it so far is also encouraging.

4. REFERENCES

[1] Agichtein, E., Lawrence, S., Gravano, L. Learning Search Engine Specific Query Transformations for Question Answering. 10th WWW Conference, 2001.

[2] Dumais, S., Banko, M., Brill, E., Lin, J., and Ng, A. Web Question Answering: Is More Always Better? ACM Conference on Information Retrieval, 2002.

[3] Radev, D., Fan, W., Qi, H., Wu, H., Grewal, A. Probabilistic Question Answering on the Web. 11th WWW Conf. 2002.

[4] Surdeanu, M., Moldovan, D. and Harabagiu, S. Performance Analysis of a Distributed Question Answering System', IEEE Transactions on Parallel and Distributed Systems, June 2002.

[5] Voorhees, E. and Harman, D., Eds. Proceedings of the Tenth Text REtrieval Conference (TREC 2001).

[6] Brill, E.. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. Computational Linguistics, 21(4):543--566, 1995.

[7] Ravichandran, D., & Hovy, E. Learning surface text patterns for a question answering system. In Proceedings of the 40th Annual Meeting of the ACL, pages 41-47.