Multiple-Engine Question Answering in TextMap

Abdessamad Echihabi, Ulf Hermjakob, Eduard Hovy,
Daniel Marcu, Eric Melz, Deepak Ravichandran
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
{echihabi,ulf,hovy,marcu,melz,ravichan}@isi.edu

1. Introduction

At TREC-2003, TextMap participated in the Main task, which encompassed answering the following types of questions:

- factoid questions;
- list questions;
- definition questions.

In this paper, we overview the architecture of the TextMap system and report its performance, as evaluated by the NIST assessors, on each of these tracks.

2. System architecture

2.1 Answering factoid and list questions

The TextMap system that answers factoid questions implements the following pipeline:

- A question analyzer identifies the expected answer type for the question given as input (see Section 3.1 for details).
- A query generator produces Web- and TREC-specific queries. The query generator exploits a database of paraphrases (see Section 3.2).
- Web queries are submitted to Google and TREC queries are submitted to the IR engine Inquery (Callan et al., 1995), to retrieve respectively 100 Web and 100 TREC documents.
- A sentence retrieval module selects 100 sentences each from the retrieved Web and TREC documents that are most likely to contain a correct answer.
- Three distinct answer selection modules (knowledge-, pattern-, and statistical-based) each pinpoint the correct answers in the resulting 200 sentences and assign them a score (see Section 3.3).
- A maximum-entropy-based re-ranker is used to combine the outputs of the three answer selection modules into a single ranked list (Section 3.4).
- If the top answer comes from a TREC sentence, the answer is presented to the user. If the top answer comes from a Web document, it is retrofitted to the TREC collection. When retrofitting fails, the system returns NIL.

List questions are answered by a system having the same architecture as the system used to answer factoid question. The only difference pertains to a simple backend algorithm that is used to output not one, but all answers whose score is higher than a given threshold.

2.2 Answering definition questions

Definition questions are answered by a system that uses many of the components described above and some additional ones. It scores answer candidates using the following resources:

- WordNet glosses
- Collection of 14,414 biographies from biography.com
- Mike Fleischman's corpus of 966,557 descriptors of proper people
- Set of subject-verb, object-verb, subject-copula-object relations

More details of the QA system that answers definition questions are given in Section 3.5.

In what follows, we describe in more detail each of the individual components and report the performance of our system.

3. Main Components

3.1 Question analysis

After parsing a question, TextMap determines its answer type, or "Qtarget", such as *PROPER-PERSON*, *PHONE-NUMBER*, or *NP*. We have built a typology of currently 185 different types, organized into several classes (Abstract, Semantic, Relational, Syntactic, etc.).

3.2 Query generation via reformulation

To bridge the gap between question and answer sentence wordings, TextMap uses paraphrasing. For any given question, TextMap generates a set of high-precision meaning-preserving reformulations to increase the likelihood of finding correct answers in texts. These reformulations are used to generate more focused TREC and Web queries. For example, the question below produces the following reformulations:

Question: How did Mahatma Gandhi die?

Reformulation patterns:

- Mahatma Gandhi died <how>?
- Mahatma Gandhi died of <what>?
- Mahatma Gandhi lost his life in <what>?
- Mahatma Gandhi was assassinated?
- Mahatma Gandhi committed suicide?
- … plus 40 other reformulations …

The fourth reformulation will easily match “Mahatma Gandhi was assassinated by a young Hindu extremist,” preferring it over alternatives such as “Mahatma Gandhi died in 1948.”

The reformulation collection in TextMap currently contains 550 assertions grouped into about 105 equivalence blocks. At run-time, the number of reformulations produced by our current system varies from one reformulation (which might just rephrase a question into a declarative form) to more than 40. On the TREC-2003 questions, we produced, on average, 5.03 reformulations per question.

3.3 Answer selection

3.3.1 Knowledge-based answer selection

The knowledge-based answer selection module uses the CONTEX parser (Hermjakob, 1997; 2001), a decision tree based deterministic parser, which has been enhanced for question answering by an additional treebank of 1,200 questions, named entity tagging that among other components uses BBN’s IdentiFinder (Bikel et al., 1999), and a semantically motivated parse tree structure that facilitates matching for paraphrasing and of question/answer pairs. Answer selection is guided by the degree of matching at the syntactic/semantic level between question and answer parse trees and several additional heuristics that penalize answers for a variety of reasons, including:

- Qtarget match factor: Q: How long did the Manson trial last? Semantic mismatch: 20 miles
- Vagueness penalty: Q: Where is Luxor? Too vague: on the other side
- Negation penalty: Q: Who invented the electric guitar? Negation: Fender did **not** invent the electric guitar.

The knowledge-based answer selection module uses a limited amount of external knowledge to enhance performance: the WordNet hierarchy; internal quantity and calendar conversion routines; and abbreviation routines. See (Echihabi et al., forthcoming) for a detailed description.

3.3.2 Pattern-based answer selection

The pattern-based answer selection module uses a set of automatically learned patterns (Ravichandran and Hovy, 2002). The learning proceeds in two steps:

1. Given a Qtarget from the TextMap ontology (a relation such as BIRTHYEAR), and a few instances of <Question; Answer> pairs such as (NAME_OF_PERSON, BIRTHYEAR), extract from the web all the different patterns (TEMPLATEs) that contain such a pair.
2. Calculate the precision of each pattern and keep the patterns of high precision.

For example, for BIRTHYEAR, some of the patterns learned by the system and the precision of those patterns are:

Prec.	**#Correct**	**#Found**	**Pattern**
1	122	122	<NAME> (<BD>- <DD>
1	15	15	<NAME> (<BD> - <DD>) ,
1	13	13	, <NAME> (<BD> - <DD>)

0.9166	11	12	<NAME> was born on <BD> in
0.9090	10	11	<NAME> : <BD> - <TIME>
0.6944	25	36	<NAME> was born on <BD>

Once a set of patterns is learned, the pattern-based answer selection system uses them in order to find possible answer candidates. The set of potential answers is sorted into a ranked list using a maximum-entropy-based framework, in the style of Ittycheriah (2002). The features used by the system are the type of pattern; the frequency of an answer; the expected answer type; the word-overlap between the question and the answer sentence. The pattern-based ranker was trained using the 1192 questions in the TREC 9 and TREC 10 data sets for training and the 500 questions in the TREC 11 data set for cross-validation.

3.3.3 Statistics-based answer selection

The statistics-based answer selection module implements a noisy-channel model for answer selection - see (Echihabi and Marcu, 2003) for details. This model explains how a given sentence S_A that contains an answer sub-string A to a question Q can be rewritten into Q through a sequence of stochastic operations. Given a corpus of question-answer pairs (Q, S_A), one can train a probabilistic model for estimating the conditional probability $P(Q \mid S_A)$. Once the parameters of this model are learned, given a question Q and the set of sentences Σ returned by the IR engine, we find the sentence $S_i \in \Sigma$ and an answer in it $A_{i,j}$ by searching for the $S_{i,A_{i,j}}$ that maximizes the conditional probability $P(Q \mid S_{i,A_{i,j}})$. The probability model is trained using off-the-shelf parameter estimation package that was developed for statistical machine translation.

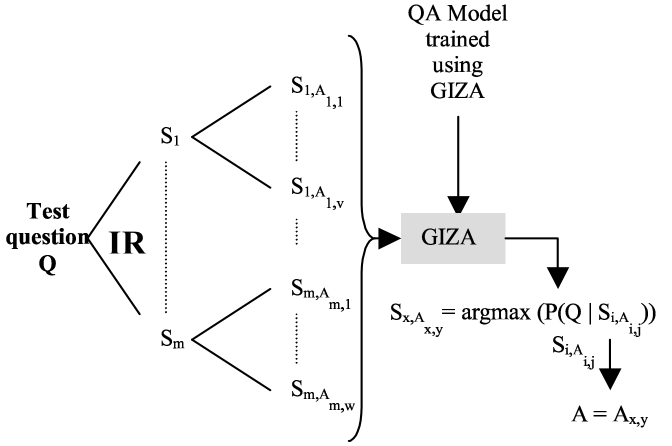

Figure 1: The noisy-channel approach to question answering.

For training, we use TREC 9 and TREC 10 questions (1091) with answer sentences (18618) automatically generated from the corresponding judgment sets. We also use questions (2000) from http://www.quiz-zone.co.uk with answer sentences (6516) semi-

automatically collected from the web and annotated for correctness by a linguist[1]. To estimate the parameters of our model, we use GIZA, a publicly available statistical machine translation package (http://www.clsp.jhu.edu/ws99/projects/mt/).
Figure 1 depicts graphically the noisy-channel based answer selection.

3.4 Combining the output of multiple answer-selection modules

Simple inspection of the outputs produced by our individual components on a development corpus of questions from the TREC-2002 collection revealed several consistent patterns of error:

- The pattern-based answer selection module performs well on questions whose Qtargets are recognizable named entities (NAMES, ORGANIZATIONS, LOCATIONS). However, this module performs less well on questions with more general QTargets (NPs, for example).
- The statistics-based answer selection module does not restrict the types of the answers it expects for a question Qtargets: it assumes that any semantic constituent can be an answer. As a consequence, many of the answers produced by this module may not be exact.
- Overall, all modules made some blatant mistakes. The pattern-based and statistics-based modules in particular sometimes select as top answers strings like “he”, “she”, and “it”, which are unlikely to be good answers for any factoid question one may imagine.

To address these problems, we decided to use the maximum entropy framework to re-rank the answers produced by the answer selection modules and root out the blatant errors. For this purpose, we have used a set of 48 feature functions, which can be classified into the following categories:

1. Component-specific: Scores from the individual answer selection module and associated features like the rank of the answer and the presence/absence of the answers produced by the individual answer selection module. We also add features based on the scores produced by the IR module and word overlap between question and answers.
2. Redundancy-specific: Count of candidate answers in the collection. We also add additional features for the logarithm and the square root of the counts.
3. Qtarget-specific: It was observed that some of the answer selection modules answer certain Qtargets better than others. We therefore model a set of features wherein we combine certain classes of Qtarget with the score of the individual answer selection module. This enables the maximum entropy model to assign different model parameters to different types of questions and answer selection module.
4. Blatant-error-specific: We model a set of features based on the development set in an iterative process. These include (negative) features such as “answers usually do not have personal pronouns” and “WHEN questions usually do not contain day of the week as answer”, among others.

---

[1] We are grateful to Miruna Ticrea for annotating the question-answer pairs.

3.5 Special modules/resources for answering definition questions

The question-answering track for TREC-2003 included a new definition subtask, in which definition questions such as “Who is Aaron Copland?”, “What is a golden parachute?”, and “What is Bausch&Lomb?” had to be answered by a set of relevant information nuggets.

Given a question like “Who is Aaron Copland?”, the challenge here is to identify relevant sentences such as

- Aaron Copland's death comes a definitive biography of America's most important composer
- Copland was born in 1900, the son of Russian Jewish immigrants (original name: Kaplan)

in a larger set of sentences that the IR module extracted as potential candidates:

- So she took me to meet Aaron Copland, who was then in his early thirties.
- Each recipient of the honor, known as the Aaron Copland Awards, will get the run of the place with a spouse or partner, but no children or pets are allowed.

We built and used several resources to identify relevant information

3.5.1 Collection of Biographies

We were able to obtain 14,414 biographical entries from http://www.biography.com, which we used to extract core biographical information for specific people as well as identify words that are indicative of biographical information.

We built a list of 6,640 words that occur at least five times in biographies and occur much more frequently in biographical text (biography.com) than in standard text (Wall Street Journal). The top 20 terms (along with their “strength”) are:

494.0 Nobel
467.5 Oxford
406.0 poems
384.0 knighted
290.0 Info
278.0 Ballet
257.0 Broadway
251.4 studied
247.0 travelled
209.0 poem
206.0 Labour
204.0 Composer
194.5 St
188.7 poetry
188.3 edited
187.5 Painter
183.0 Angeles
181.7 Physicist
171.9 War
169.2 commanded

Ten terms of lesser strength that nevertheless boost the score in the Copland examples:

49.5 Symphony
47.4 teacher
42.5 Music
34.7 Oscar
22.3 son
18.7 achievements
16.5 immigrant
13.2 established
12.9 advocate
10.1 Jewish

The second use of the biography collection was to boost answer candidates with words from core biography terms for the specific person in question:

- Copland , Aaron 1900 -- 1990 is a Composer; born in New York City.

3.5.2 Collection of Descriptors for Proper People

TextMap uses a slightly cleaned-up version of Michael Fleischman's list of 966,557 descriptor entries for proper people (Fleischman et al., 2003) such as
10 composer , composer , composer , Aaron Copland
2 witness , witness , witness , Aaron Copland
1 musician , musician , Pulitzer Prize winning musician , Aaron Copland
1 musician , musician , musician , Aaron Copland
1 composer , composer , late composer , Aaron Copland
1 composer , composer , classical composer , Aaron Copland
boosting the score of sentences containing descriptors such as "composer" or "Pulitzer Prize winning musician".

3.5.3 WordNet

If the anchor term of the definition question (e.g. "Aaron Copland") has a WordNet gloss (Miller and Fellbaum, 1998), TextMap gives credit for sentences that contain words and expressions that overlap with that WordNet gloss (as underlined in the example below):

WordNet gloss: Copland, Aaron Copland -- United States composer (1900-1990)

3.5.4 Semantic-Relationship Patterns

TextMap gives credit when the anchor of a question is found to match one of currently 110 semantic relations, for example when the question anchor is the logical subject of verbs like "to compose", "to write", "to teach"; when it is the logical object of verbs like "to bear" (to be born); when it is in a subject-copula-object relationship with a head noun like "composer", "advocate", "son" (or any other relative).

- Aaron Copland composed "Fanfare for the Common Man."
- Aaron Copland was born in 1900, the son of Russian Jewish immigrants

3.5.5 Duplicate Removal and Answer Cutoff

Duplicates are avoided by giving credit only for those words, expressions and relations that have not yet been observed in higher-scoring information nuggets.

To determine the number of answer nuggets for a given question, we start with a very low threshold score and then linearly increase the threshold for each answer, eliminating answers with a score lower than the rising threshold. This mechanism keeps the number of answers in a reasonable range while allowing more answers for questions with many high-scoring answer candidates. This allowed our system to provide over 20 nuggets for questions such as “Who is Bill Bradley/Aaron Copland/Andrew Carnegie?” while limiting itself to as few as 3 answers questions such as “What is Iqra/El Shaddai/ Restorative Justice?” for which the system found fewer promising information nuggets.

For the definition questions in TREC-2003, our system returned an average of 12.2 answers and 1663.1 bytes per question.

4. Evaluation Results

In the official TREC evaluation, TextMap scored 33.7% for factoid questions, 11.8% for list questions, and 46.1% for definition questions, resulting in a composite score of 31.3%

We also performed the following three sets of experiments with the Web as our corpus.

1. Maximum Entropy re-ranking performed on individual answer selection modules: In this experiment, we turn off all the features in any answer selection module that depends on another answer selection module and we use for re-ranking only the top 50 answers supplied by one answer selection component. Thus this experiment enables us to assess the impact separately on each individual answer selection module of redundancy-, Qtarget-, and blatant-error-specific features, as well as improved weighting of the features specific to each module.
2. Maximum Entropy re-ranking on all answer selection modules: In this experiment we add all the features described in Section 5.3 and test the performance of the combined system after re-ranking. Re-ranking is performed on the answer set that results from combining the top 50 answers returned by each individual answer selection module. This experiment enables us to assess whether the ME framework enables us to better exploit strengths specific to each answer selection module.
3. Feature Selection: Since we have only 200 training examples and almost 50 features, the training is likely to be highly prone to over-fitting. To avoid this problem we apply feature selection as described in (Della Pietra et al., 1996). This reduces the number of features from 48 to 31.

Metric	Knowledge-Based		Pattern-Based		Statistical-Based		Base from all systems followed by ME re-ranking (no feature selection)	Base from all systems followed by ME re-ranking (with feature selection)
	Base	Base followed by ME re-ranking	Base	Base followed by ME re-ranking	Base	Base followed by ME re-ranking		
Top answer	35.83%	45.03%	25.18%	30.50%	21.30%	32.20%	46.37%	47.21%
Top 5	57.38%	56.41%	35.59%	43.09%	31.23%	40.92%	57.62%	57.62%
MRR	43.88	49.36	28.57	35.37	24.83	35.51	51.07	51.27

Table 1. Performance of various answer selection modules in TextMap, an end-to-end QA system.

Table 1 summarizes the results: it shows the percentage of correct, exact answers returned by each answer selection module with and without ME-based re-ranking, as well as the percentage of correct, exact answers returned by an end-to-end QA system that uses all three answer selection modules together. Table 1 also shows the performance of these systems in terms of percentage of correct answers ranked in the top 5 answers and the corresponding MRR scores.

The results in Table 1 show that appropriate weighting of the features used by each answer selection module as well as the ability to capitalize on global features, such as the counts associated with each answer, are extremely important means for increasing the overall performance of a QA system. ME re-ranking led to significant increases in performance for each answer selection module individually. When measured with respect to the ability of our system to find correct, exact answers when returning only the top answer, it accounted for 14.33% reduction in the error rate of the knowledge-based answer selection module; 7.1% reduction in the error rate of the pattern-based answer selection module; and 13.85% reduction in the error rate of the statistical-based answer selection module. Combining the outputs of all systems yields an additional increase in performance; when over-fitting is avoided through feature selection, the overall reduction in error rate is 3.96%. This corresponds to an increase in performance from 45.03% to 47.21% on the top-answer accuracy metric.

The inspection of the weights computed by the ME-based feature selection algorithm shows that our log-linear model was able to lock and exploit strengths and weaknesses of the various modules. For example, the weight learned for a feature that assesses the likelihood of the pattern-based module to find correct answers for questions that have QUANTITY as a Qtarget is negative, which suggests that the pattern-based module is not good at answering QUANTITY-type questions. The weight learned for a feature that assesses the likelihood of the statistics-based module to find correct answers for questions that have an UNKNOWN or NP Qtarget is large, which suggests that the statistical module is better suited for answering these types of questions. The knowledge-based module is better than the others in answering QUANTITY and DEFINITION questions. These results are quite intuitive: The pattern-based module did not have enough QUANTITY-type questions in the training corpus to learn any useful patterns. The Statistics-based module implemented here does not explicitly use the Qtarget in answer selection and does not model the source probability of a given string being a correct answer. As a consequence, it explores a much larger space of choices when determining an answer compared to the other two modules.

At the moment, the knowledge-based module yields the best individual results. This is not surprising given that it is a module that was engineered carefully, over a period of three years, to accommodate various types of Qtargets and knowledge resources. Many of the resources used by the knowledge-based module are not currently exploited by the other modules: the semantic relations identified by CONTEX; the ability to exploit the paraphrase patterns and advanced forms of reformulation for answer pinpointing; the external sources of knowledge (WordNet; abbreviation lists; etc.); and a significant set of heuristics (see Section 2.4).

Our results suggest that in order to build good answer selection modules, one needs to both exploit as many sources of knowledge as possible and have good methods for integrating them. The sources of knowledge used only by the knowledge-based answer selection module proved to have a stronger impact on the overall performance of our answer selection systems than the ability to automatically train parameters in the pattern- and statistics-based systems, which use poorer representations. Yet, the ability to properly weight the contribution of various knowledge resources was equally important. For example, Maximum Entropy naturally integrated additional features into the knowledge-based answer selection module; a significant part of the 9.2% increase in correct answers reported in Table 1 can be attributed to the addition of redundancy features, a source of knowledge that was unexploited by the base system.

References

Bikel, D., R. Schwartz, and R. Weischedel. 1999. An Algorithm that Learns What's in a Name. *Machine Learning—Special Issue on NL Learning*, 34, 1–3.

Callan, J.P., W.B. Croft, and J. Broglio. 1995. "TREC and Tipster Experiments with Inquery. *Information Processing and Management* 31(3), 327-343.

Echihabi, A., U. Hermjakob, E. Hovy, D. Marcu, E. Melz, D. Ravichandran. 2003. How to Select an Answer String. *Forthcoming*.

Echihabi, A., and D. Marcu. 2003. A Noisy-Channel Approach to Question Answering. *Proceedings of the 41st Meeting of the Association for Computational Linguistics (ACL)*, July 7-12, Sapporo, Japan.

Fellbaum, C. (Editor); G. Miller (Preface). 1998. WordNet: An Electronic Lexical Database. MIT Press. http://www.cogsci.princeton.edu

Fleischman, M., E. Hovy and A. Echihabi. 2003. Offline Strategies for Online Question Answering: Answering Questions Before They Are Asked. *Proceedings of the 41st Meeting of the Association for Computational Linguistics (ACL)*, July 7-12, Sapporo, Japan.

Hermjakob, U. 1997. *Learning Parse and Translation Decisions from Examples with Rich Context.* Ph.D. dissertation, University of Texas Austin. file://ftp.cs.utexas.edu/pub/mooney/papers/hermjakob-dissertation-97.ps.gz.

Hermjakob, U., A. Echihabi, and D. Marcu. 2002. Natural Language Based Reformulation Resource and Web Exploitation for Question Answering. *Proceedings of the TREC-11.* NIST, Gaithersburg, MD.

Hovy, E.H., U. Hermjakob, C.-Y. Lin, and D. Ravichandran. 2002. Using Knowledge to Facilitate Pinpointing of Factoid Answers. *Proceedings of the COLING-2002 Conference.* Taipei, Taiwan.

Ittycheriah, A. 2001. *Trainable Question Answering System*. Ph.D. Dissertation, Rutgers, The State University of New Jersey, New Brunswick, NJ.

Ravichandran, D. and E.H. Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL).* Philadelphia, PA 41–47.