# Question Answering in Webclopedia

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, Chin-Yew Lin

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292-6695
Tel: 310-448-8731
Fax: 310-823-6714
Email: {hovy,gerber,ulf,junk,cyl}@isi.edu

## 1. Introduction: Question Answering

IR techniques have proven quite successful at locating within large collections of documents those relevant to a user's query. Often, however, the user wants not whole documents but brief answers to specific questions: *How old is the President? Who was the second person on the moon? When was the storming of the Bastille?* Recently, a number of research projects have investigated the computational techniques needed for effective performance at this level of granularity, focusing just on questions that can be answered in a few words taken as a passage directly from a single text (leaving aside, for the moment, the answering of longer, more complex answers, such as stories about events, descriptions of objects, compare&contrast discussions, arguments of opinion, etc.).

The systems being built in these projects exhibit a fairly standard structure: all create a query from the user's question, perform IR with the query to locate (segments of) documents likely to contain an answer, and then pinpoint the most likely answer passage within the candidate documents. The most common difference of approach lies in the pinpointing. A 'pure IR' approach would segment each document in the collection into a series of mini-documents, retrieve the segments that best match the query, and return them as answer. The challenge here would be to make segments so small as to be just answer-sized but still large enough to be indexable. A 'pure NLP' approach would be to match the parse and/or semantic interpretation of the question against the parse and/or semantic interpretation of each sentence in the candidate answer-containing documents, and return the best match(es). The challenge here would be to perform parsing, interpretation, and matching fast enough to be practical, given the large volumes of text to be handled.

Answering short questions thus becomes a problem of finding the best combination of word-level (IR) and syntactic/semantic-level (NLP) techniques, the former to produce as short a set of likely candidate segments as possible and the latter to pinpoint the answer(s) as accurately as possible.

Because language allows paraphrasing and inference, however, working out the details is not entirely straightforward. In this paper we describe the Webclopedia, a system that uses a classification of QA types to facilitate coverage, uses a robust syntactic-semantic parser to perform the analysis, and contains a matcher that combines word- and parse-tree-level information to identify answer passages. Section 2 outlines the Webclopedia approach and architecture; Section 3 describes document retrieval and processing, Section 4 describes the QA Typology, Section 5 the parsing, and Section 6 the matching.

## 2. Webclopedia

Webclopedia's architecture, shown in Figure 1, follows the pattern outlined in Section 1:

**Parsing of question**: The CONTEX parser (see Section 5) is used to parse and analyze the question, assisted by BBN's IdentiFinder (Bikel et al., 1999).

**Question analysis**: To form a query, single- and multi-word units (content words) are extracted from the parsed query. WordNet synsets are used for query expansion. See Section 3.

**IR**: The IR engine MG (Witten et al. 1994) is used to return and rank the top 1000 documents.

**Segmentation**: To decrease the amount of text to be processed, the documents are broken into semantically coherent segments. Two text segmenters were tried—TexTiling (Hearst, 94), C99 (Choi, 00); the first is used.

**Ranking of segments**: For each segment, each sentence is scored using a formula that rewards word and phrase overlap with the question and expanded query words. The segments are ranked.

**Parsing of segments**: CONTEX also parses each sentence of the top-ranked 100 segments.

**Pinpointing**: For each sentence, three steps of matching are performed (see Section 6); two compare the parses of the question and the sentence; the third moves a fixed-length window over each sentence and computes a goodness score based on the words and phrases contained in it.

**Ranking of answers**: The candidate answers' scores are compared and the winning answer(s) are output.

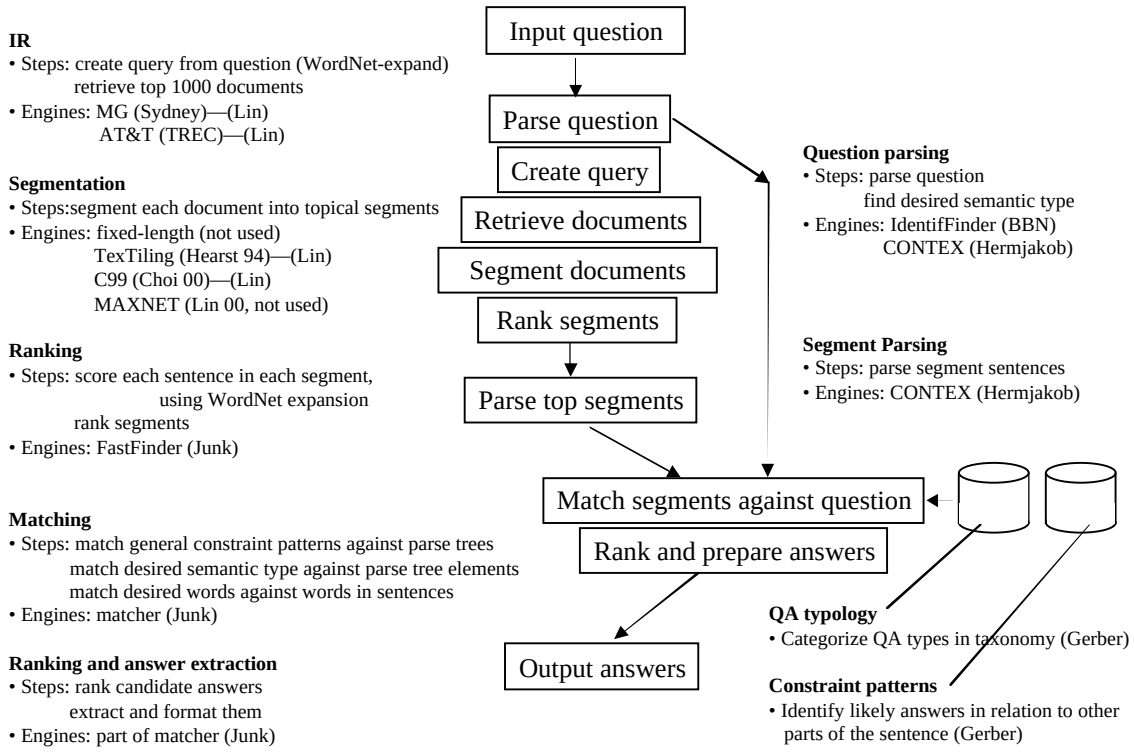


Figure 1. Webclopedia architecture.

## 3. Information Retrieval and Document Ranking

### Analyzing the Question to Create a Query

We parse input questions using CONTEX (Section 5) to obtain a semantic representation of the questions. For example, we determine that the question "Who is Johnny Mathis' high school track coach?" is asking for the name of person. The question analysis module identifies noun phrases, nouns, verb phrases, verbs, adjective phrases, and adjectives embedded in the question. These phrases/words are assigned significance scores according to the frequency of their type in our question corpus (a collection of 27,000+ questions and answers), secondarily by their length, and finally by their significance scores, derived from word frequencies in the question corpus.

We remain indebted to BBN for the use of IdentiFinder (Bikel et al., 1999), which isolates proper names in a text and classifies them as person, organization, or location.

### Expanding Queries

In order to boost recall we use WordNet 1.6 (Fellbaum 1998) to expand query terms and place all the expanded terms into a Boolean expression. For example, "high school" is expanded to:

"(high&school)|(senior&high&school)|(senior&high)|high|highschool"

It is obvious that such brute force expansion has undesirable effects. The expanded "high school" query contains "high". This will make "high school" relatively less significant, since "high" is a very common word. We did not try to fix this problem in this year's TREC evaluation, but are planning to improve the expansion procedure next year.

### Retrieving Documents

We use MG (Witten et al. 1994) as our search engine. Although MG is capable of performing ranked query, we only use its Boolean query capability. For the entire TREC9 test corpus, the size of the inverse index file is about 200 MB and the size of the compressed text database is about 884 MB. The stemming option is turned on. Queries are sent to the MG database, and the retrieved documents are ranked according to their ranking from query analysis. For example:

Johnny&mathis&((high&school)|(senior&high&school)|(senior&high)|high|highschool)

will be sent to the database first. If the number of documents returned is less than a pre-specified threshold then we retain this set of documents as the basis for further processing. The threshold is set to 5,000 in our TREC9 evaluation. If nothing is returned then we relax the query by taking the next query term in our query rank list. In this case, it is "high school track coach". If more than 5,000 documents are returned we drop the query expansion and use the original query terms instead. For this example, the query will be "Johnny&mathis&high&school&track&coach".

In some cases, it is impossible to get the number of returned documents down to 5,000. For example, the question "What is the meaning of life?" will return an enormous amount of documents since all the words in the query are very common. We plan to address this problem by adding proximity and order constraints to the query process.

### Ranking Documents

If the total numbers of documents returned by MG is N, we would like to rank the documents to maximize answer recall and precision in the topmost K << N, in order to minimize the parsing

and subsequent processing. In this phase we set K=1,000. Our document ranker uses the following scoring method:

- Each question word gets a score of 2
- Each synonym gets a score of 1
- Other words get a score of 0

Normally common words are ignored unless they are part of a phrase in question word order, in which case they get a score of 2 along with other words in the phrase. Based on these scores, the total score for a document is:

Document score = sum of word scores / number of different words

## Segmenting Documents

Splitting each document into topical segments to be input to the matcher is based on the assumption that important contextual information for pinpointing answers tends to occur within a local context. This is mostly true for the setup of TREC9 Q&A. Furthermore, CONTEX does not use information outside sentence boundaries. This step helps the system focus on smaller regions of text where answers are most likely to be found.

We tried two text segmenters, TextTiling (Hearst 1994) and C99 (Choi 2000). They perform at almost the same level, though TextTiling is faster.

## Ranking Segments

The resulting segments are re-ranked using the same ranker described earlier. This time, only the topmost 100 segments are passed to the parser (and then to the matcher for answer pinpointing).

## Retrieval Results

We evaluated our IR front end in 6 separate experiments using the 238 training questions obtained from NIST. The resulting answer distributions within the top 1,000 segments are shown in Table 1.

| N <= | 5 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 500 | 1000 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Test0=38 | 12 | 15 | 22 | 23 | 25 | 27 | 28 | 28 | 29 | 29 | 30 | 36 | 36 |
| % | 19% | 23% | 34% | 36% | 39% | 42% | 44% | 44% | 45% | 45% | 47% | 56% | 56% |
| Test1=52 | 27 | 31 | 38 | 39 | 41 | 41 | 41 | 41 | 41 | 41 | 41 | 48 | 48 |
| % | 42% | 48% | 59% | 61% | 64% | 64% | 64% | 64% | 64% | 64% | 64% | 75% | 75% |
| Test2=64 | 23 | 29 | 38 | 41 | 43 | 46 | 47 | 47 | 47 | 48 | 48 | 56 | 56 |
| % | 36% | 45% | 59% | 64% | 67% | 72% | 73% | 73% | 73% | 75% | 75% | 88% | 88% |
| Test3=52 | 17 | 21 | 24 | 27 | 29 | 30 | 31 | 32 | 32 | 32 | 32 | 44 | 46 |
| % | 33% | 40% | 46% | 52% | 56% | 58% | 60% | 62% | 62% | 62% | 62% | 85% | 89% |
| Test4=55 | 34 | 39 | 46 | 48 | 48 | 50 | 50 | 50 | 50 | 51 | 51 | 54 | 54 |
| % | 62% | 71% | 84% | 87% | 87% | 91% | 91% | 91% | 91% | 93% | 93% | 98% | 98% |
| Test5=54 | 25 | 30 | 34 | 38 | 40 | 41 | 42 | 43 | 43 | 45 | 45 | 51 | 51 |
| % | 46% | 56% | 63% | 70% | 74% | 76% | 78% | 80% | 80% | 83% | 83% | 94% | 94% |
| Overall | 138 | 165 | 202 | 216 | 226 | 235 | 239 | 241 | 242 | 246 | 247 | 289 | 291 |
| % | 44% | 52% | 64% | 69% | 72% | 75% | 76% | 77% | 77% | 78% | 78% | 92% | 92% |

Table 1. Percentage of topmost N segments containing an answer after retrieval and ranking.

It is interesting to see that the system gets about 52% of answer segments within the top 10 and reaches only 78% within the top 100. And even in the top 1000 segments, 8% of the answers are missing. This indicates that further improvement of the IR front end is critical.

## 4. The QA Typology

There are many ways to ask the same thing. Likewise, there are many ways of delivering the same answer. Such variations form a sort of semantic equivalence class of both questions and answers; speaking approximately, any form of the question can be answered by any form of the answer. Since the user may employ any version of his or her question, and the source documents may contain any version(s) of the answer, an efficient system should group together equivalent question types and answer types. Any specific question can then be indexed into its type, from which all equivalent forms of the answer can be ascertained. These QA equivalence types can help with both query expansion (for IR) and answer pinpointing (for NLP).

However, the equivalence is fuzzy; even slight variations introduce exceptions: *who invented the gas laser?* can be answered by both *Ali Javan* and *a scientist at MIT*, while *what is the name of the person who invented the gas laser?* requires the former only. This inexactness suggests that the QA types be organized in an inheritance hierarchy, allowing the answer requirements satisfying more general questions to be overridden by more specific ones 'lower down'.

Previous work in automated question answering has often categorized questions by question word alone or by a mixture of question word and the semantic class of the answer (Srihari and Li, 2000; Moldovan et al., 2000). To ensure full coverage of all forms of simple question and answer, we have been developing a QA Typology as a taxonomy of QA types, becoming increasingly specific as one moves from root downward. Instead of focusing on question word or semantic type of the answer, our classes attempt to represent the user's intention, including for example the classes Why-Famous (for *Who was Christopher Columbus?* but not *Who discovered America?*, which is a Proper-Person QA type) and Abbreviation-Expansion (for *What does NAACL stand for?*).

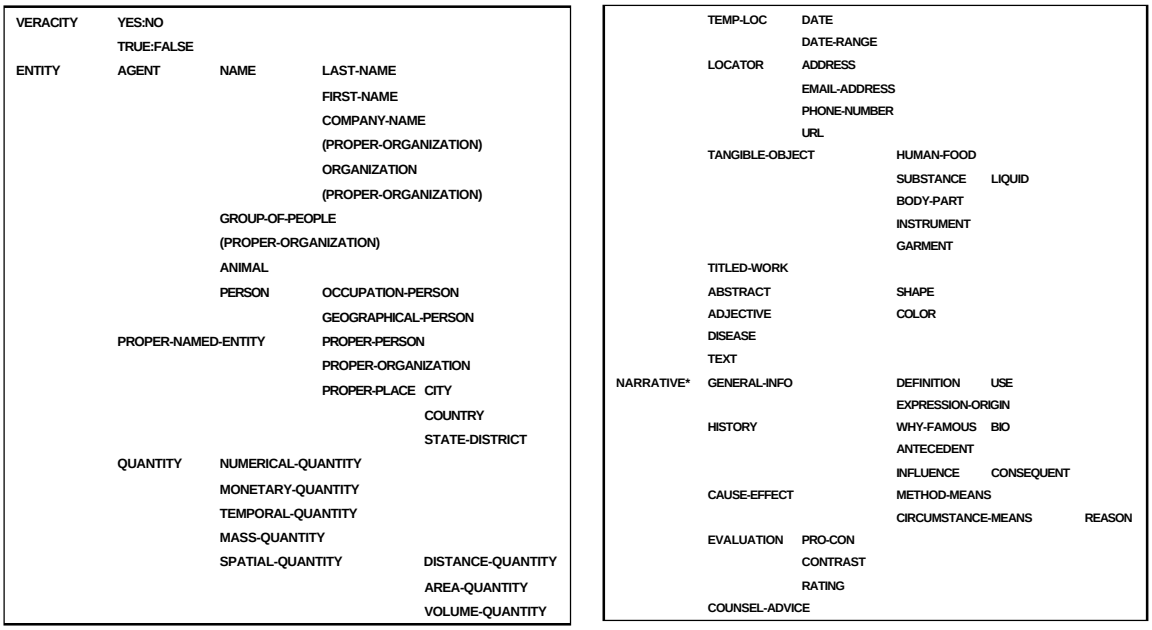


Figure 2. Portion of Webclopedia QA Typology.

To create the QA Typology, we analyzed 17,384 questions and their answers (downloaded from answers.com); see (Gerber, 2001). The Typology contains 94 nodes, of which 47 are leaf nodes; a section of it appears in Figure 2.

Each Typology node has been annotated with examples and typical patterns of expression of both Question and Answer, as indicated in Figure 3 for Proper-Person.

| **Question examples** | **Question templates** |
|---|---|
| Who was Johnny Mathis' high school track coach? | who be <entity>'s <role> |
| Who was Lincoln's Secretary of State? | |
| Who was President of Turkmenistan in 1994? | who be <role> of <entity> |
| Who is the composer of Eugene Onegin? | |
| Who is the CEO of General Electric? | |

| **Actual answers** | **Answer templates** |
|---|---|
| Lou Vasquez, track coach of…and Johnny Mathis | <person>, <role> of <entity> |
| Signed Saparmurad Turkmenbachy [Niyazov], president of Turkmenistan | <person> <role-title*> of <entity> |
| ...Turkmenistan's President Saparmurad Niyazov… | <entity>'s <role> <person> |
| ...in Tchaikovsky's Eugene Onegin... | <person>'s <entity> |
| Mr. Jack Welch, GE chairman... | <role-title> <person> ... <entity> <role> |
| ...Chairman John Welch said ...GE's | <subject>\|<psv object> of related role-verb |

Figure 3. Portion of QA Typology node annotations for Proper-Person.

## 5. Parsing

Some answers returned by a youthful Webclopedia showed the need to ensure that the answer found is of the right kind semantically:

Q: Where are zebras most likely found? — A: in the dictionary

Q: Where do lobsters like to live? — A: on the table / at the same rate as regular lobsters

and in the right range numerically:

Q: How many people live in Chile? — A: nine

We use CONTEX, a parser that is trained on a corpus to return both syntactic and semantic information, to help.

CONTEX is a deterministic machine-learning based grammar learner/parser that was originally built for MT (Hermjakob, 1997; Hermjakob and Mooney, 1997), where a smaller version of CONTEX (lexically restricted English) reached a labeled precision rate of 89.8% when trained on 256 sentences. Over the past few years it has been extended over the past years to handle deployment on new languages, including Japanese and Korean (Hermjakob, 2000). The Japanese version of the parser, trained on 4096 sentences and tested on lexically unrestricted sentences, achieves 91.4% labeled precision and 91.1% labeled recall for parse trees with a word level granularity, and a bunsetsu level dependency accuracy rate of 84.5%. For English, CONTEX parses of unseen sentences measured 87.6% labeled precision and 88.4% labeled recall, after being trained on 2048 sentences from the Penn Treebank in March 2000. The robustness and the

fact that the parser produces a complete parse tree for every test sentence, makes it very useful for Webclopedia.

CONTEX works as follows. As with statistical systems, the grammar learning system also induces its rules from training data; however, it makes better use of linguistic knowledge and other knowledge resources. When presented with a set of parse trees, the learning system automatically derives the sequence of Shift-Reduce parsing operations required to produce each tree. To determine which specific action to take at any point, it considers features of the left and right contexts of the current word. These features include words, parts of speech, lexical and semantic features, etc. In cases of ambiguity, it asks the trainer to identify which feature(s) to pay attention to. Viewing ambiguity as a decision making problem, the system builds a variant of a decision tree to handle the ambiguity in future, using the feature(s) within the context as well as background knowledge in the form of lexicons, ontologies, and any results from topic detection, etc. The appropriate features are indicated manually, by the trainer, if he or she decides they are needed. The decision structure however differs from a traditional grammar in two ways: (1) it is more, in the sense that it does not only provide a space of possible analyses, but in fact selects what it believes is the best analysis, and (2) it has a very operational character in that it directly drives the shift-reduce parser. The grammar as represented by the decision structure therefore has a somewhat different character from the traditional static grammar resource.

Manual guidance allows CONTEX to require far fewer treebanked sentences for training. CONTEX derives much of its strength from the integration of different types of background knowledge, even if those knowledge resources are incomplete. In this way it is a good example of the hybridization of statistical and symbolic techniques. Machine learning algorithms automatically select the most relevant features that best support specific run time parse decisions. This approach employs human and machine each to best advantage: linguists are good at parsing individual sentences, but less good at keeping all the complexity and generalization of a full grammar under control, while machines are excellent at managing and generalizing large sets of individual data points. The result is a rapid traversal of the learning space toward a robust, wide-coverage grammar and parser.

Webclopedia required four extensions to CONTEX.

First, the grammar had to be extended to handle questions. This was achieved by adding approx. 250 manually parsed questions to the Penn Treebank, on which the system's English grammar has been trained. Of these questions, 100 were obtained from NIST's TREC-8 QA corpus and 150 from elsewhere.

Second, the semantic type ontology in CONTEX was extended, both to include QA types and to include many more Objects from our ontology SENSUS. It now contains about 10,000 nodes.

Third, the results of BBN's IndentiFinder locating proper names had to be taken into account.

Fourth, the parse tree output had to be augmented to carry question-related information. The semantic type of the desired answer, as determined by CONTEX, we call the Qtarget. CONTEX returns a ranked list of Qtargets, in order of specificity, drawn from its ontology. For example, the expression

(((c-date) (c-temp-loc-with-year)) ((eq c-temp-loc)))

indicates that the system should try to match a specific date or specific year (both first choice) over a more general temporal expression like "after the war".)

Beside the Qtargets that refer to concepts in CONTEX's concept ontology (see first example in Figure 4), Qtargets can also refer to part of speech labels (see first example), to constituent roles or slots of parse trees (see second and third examples), and to more abstract nodes in the QA

Typology (see later examples). For questions with the Qtargets Q-WHY-FAMOUS, Q-WHY-FAMOUS-PERSON, Q-SYNONYM, and others, the parse tree also provides a slot Qargs that contains additional information helpful for matching (see final examples).

**Semantic ontology types** (I-EN-CITY) **and part of speech labels** (S-PROPER-NAME):

What is the capital of Uganda?
QTARGET: (((I-EN-CITY S-PROPER-NAME)) ((EQ I-EN-PROPER-PLACE)))

**Parse tree roles**:

| | |
|---|---|
| Why can't ostriches fly? | QTARGET: (((ROLE REASON))) |
| Name a film in which Jude Law acted. | QTARGET: (((SLOT TITLE-P TRUE))) |

**QA Typology nodes**:

| | |
|---|---|
| What are the Black Hills known for? | Q-WHY-FAMOUS |
| Who was Whitcomb Judson? | Q-WHY-FAMOUS-PERSON |
| What is Occam's Razor? | Q-DEFINITION |
| A corgi is a kind of what? | Q-DEFINITION |
| What is another name for nearsightedness? | Q-SYNONYM |
| Aspartame is also called what? | Q-SYNONYM |
| Should you exercise when you're sick? | Q-YES-NO-QUESTION |
| True or false: Chaucer was an actual person. | Q-TRUE-FALSE-QUESTION |

**Qargs for additional information**:

Who was Betsy Ross? QTARGET: (((Q-WHY-FAMOUS-PERSON))) QARGS: (("Betsy Ross"))
How is "Pacific Bell" abbreviated? QTARGET: (((Q-ABBREVIATION))) QARGS: (("Pacific Bell"))
What are geckos? QTARGET: (((Q-DEFINITION))) QARGS: (("geckos" "gecko") ("animal"))

Figure 4. QA-related information, returned in the parse tree of the question.

## 6. Answer Matching

Given the instantiated QA patterns, the Qtargets and Qargs lists, and the potential answer-bearing text segments (also parsed by CONTEX), the Matcher module performs three attempts to pinpoint the answer:

- match QA patterns,
- match Qtargets and Qargs,
- (if all else fails) move a word-level window across the (unparsed) text, scoring each position.

The window scoring function is as follows:

$$\text{Score} = (500 / (500+w))*(1 / r) * \Sigma[(\Sigma I^{1.5*e*u*b*q})^{1.5}]$$

Factors:

- *w*: window width (modulated by gaps of various lengths: "white house" ≠"white car and house")
- *r*: rank of Qtarget in list returned by CONTEX
- *I*: window word information content (inverse log freq)
- *q*: # different question words, and specific rewards (bonus *q=3.0*)
- *e*: penalty for question word expansion using WordNet synsets (*e=0.8*)
- *b*: boosting for main verb match, target words, proper names, etc. (*b=2.0*)
- *u*: (value 0 or 1) indicates whether a word has been "subsumed" by the Qtarget model and should not contribute (again) to the score. For example, "In what year did Columbus discover America?" the subsumed-words are {what, year}.

Unless required, we will not try to develop a more sophisticated scoring function, preferring to focus on the modules that employ information 'deeper' than the word level.

## 7. Experiments and Results

We entered the TREC-9 short form QA track, and received an overall Mean Reciprocal Rank score of 0.318, which put Webclopedia in essentially tied second place with two others. (The winning system far outperformed all the others.)

A sample analysis of the relative performance of the three modules appears in Table 2. It is clear that the QA patterns made only a small contribution, and that the Qtarget made by far the largest contribution. Interestingly, the word-level window match lay somewhere in between.

| Date | IR hits | QA pattern | Qtarget | Window | Total |
|---|---|---|---|---|---|
| 6/17 | 78.1 | 05.5 | 26.2 | 10.4 | 30.3 |

Table 2. Correct answers attributable to each module.

We are pleased with the performance of Qtargets. They indicate the value of trying to locate the desired semantic type from the meaning of the question. Together with the parse structure, they also help with pinpointing the answer closely: our average answer window length was approx. 25 bytes.

We are not however satisfied with the manually built QA patterns. First, it is too difficult and takes too long to build them by hand (the 500 we have were assembled by simply combining approx. 25 question patterns with 25 answer patterns). Second, the patterns are not robust in the face of small variations of phrasing. We aim instead to build the QA patterns automatically.

## 8. 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.

Choi, F.Y.Y. 2000. Advances in independent linear text segmentation. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics* (ANLP-NAACL-00), 26–33.

Fellbaum, Ch. (ed). 1998. *WordNet: An Electronic Lexical Database.* Cambridge: MIT Press.

Gerber, L. 2001. A QA Typology for Webclopedia. In prep.

Hearst, M.A. 1994. Multi-paragraph segmentation of expository text. *Proceedings of the* Annual Meeting of the Association for Computational Linguistics (ACL-94).

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

Hermjakob, U. and R.J.Mooney. 1997. Learning Parse and Translation Decisions from Examples with Rich Context. In *35th Proceedings of the Conference of the Association for Computational Linguistics* (*ACL)*, 482–489. file://ftp.cs.utexas.edu/pub/mooney/papers/contex-acl-97.ps.gz.

Hermjakob, U. 2000. Rapid Parser Development: A Machine Learning Approach for Korean. In *Proceedings of the North American chapter of the Association for Computational Linguistics* (NAACL-2000). http://www.isi.edu/~ulf/papers/kor_naacl00.ps.gz.

Moldovan, D., S. Harabagiu, M. Pasca, R. Mihalcea,, R. Girju, R. Goodrum, and V. Rus. 2000. The Structure and Performance of an Open-Domain Question Answering System. In *Proceedings of the Conference of the Association for Computational Linguistics* (ACL-2000), 563–570.

Srihari, R. and W. Li. 2000. A Question Answering System Supported by Information Extraction. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics* (ANLP-NAACL-00), 166–172.

Witten, I.H., A. Moffat, and T.C. Bell. 1994. Managing Gigabytes: Compressing and indexing documents and images. New York: Van Nostrand Reinhold.