Knowledge-Based Question Answering

Ulf HERMJAKOB, Eduard H. HOVY, and Chin-Yew LIN
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292-6695
USA
tel: 310-448-8731
email: {ulf,hovy,cyl}@isi.edu

ABSTRACT

This paper describes the Webclopedia Question Answering system, in which methods to automatically learn patterns and parameterizations are combined with hand-crafted rules and concept ontologies. The source for answers is a collection of 1 million newspaper texts, distributed by NIST. In general, two kinds of knowledge are used by Webclopedia to answer questions: knowledge about language and knowledge about the world. The former is embodied both in the Information Retrieval engine that identifies likely answer sentences and the CONTEX parser that analyses the input question and candidate answers. The latter is embodied in a QA typology of some 140 nodes, a concept ontology of some 10,000 concepts, and the IR engine's ranking algorithm that takes typical document structure into account.

Keywords: Question Answering, Question/Answer Typology, Qtarget, Webclopedia, CONTEX.

1. INTRODUCTION

Several research projects have recently investigated the problem of automatically answering simple questions that have brief phrasal answers ('factoids'), by identifying and extracting the answer from a large collection of text. 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 Queen of England? Who was the second person in space? When was the storming of the Bastille?* In order to pinpoint the answers within likely documents, however, more fine-grained methods have to be used. As a result, most QA systems 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 more accurately the IR engine can select passages, the less additional knowledge and/or processing is required for pinpointing.

Answering short questions then 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. Typically, IR techniques tend to use broad-brush (statistical) techniques while NLP techniques tend to use more pointed handcrafted rules.

To improve performance, QA systems are increasingly augmenting their general-purpose rules with knowledge about language and about the world.

In this paper we describe Webclopedia, a system built over the past 2-1/2 years that includes 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.

Knowledge about language is employed especially by the CONTEX parser. This knowledge is incorporated in three resources: a 30,000-node concept hierarchy, a taxonomy of 140 question/answer types, and a lexicon. With this knowledge, CONTEX is able to assign semantic roles such as Logical-Subject and Logical-Object to parts of sentences, thereby enabling the Webclopedia QA matcher to determine that in "Belli's clients have included Jack Ruby, who killed John F. Kennedy assassin Lee Harvey Oswald, and Jim and Tammy Bakker." Ruby was the killer not of Kennedy but of Kennedy's assassin, and that it was Oswald who killed Kennedy.

Knowledge about the world is employed by both CONTEX and Webclopedia's retrieval module that locates candidate answers. Currently this knowledge is incorporated primarily in WordNet. Using WordNet, for example, the system can determine directly that Belgrade is the capital of Yugoslavia, and can then score

higher candidate answer passages that contain "Belgrade". Similarly, for definition-type questions such as "what is bandwidth?" the system uses WordNet to extract words used in the term definitions, and boosts candidate answer scores appropriately. We plan soon to also use the web to determine the answer, prior to finding the answer again within the QA source corpus for justification.

2. ARCHITECTURE

Webclopedia adopts the by now more or less standard QA system architecture, namely question analysis, document / passage retrieval, passage analysis for matching against the question, and ranking of results. Its architecture (Figure 1) contains the following modules, which are described in more detail in [9,10]:

- **Question parsing:** Using BBN's IdentiFinder [2], the CONTEX parser produces a syntactic-semantic analysis of the question and determines the QA type.
- **Query formation**: Single- and multi-word units (content words) are extracted from the analysis, and WordNet synsets are used for query expansion. A series of Boolean queries is formed.
- **IR**: The IR engine MG [16] returns the top-ranked *N* documents.
- **Selecting and ranking sentences**: For each document, the most promising *K<<N* sentences are located and scored using a formula that rewards word and phrase overlap with the question and its expanded query words. Results are ranked.
- **Parsing segments**: CONTEX parses the top-ranked 300 sentences.
- **Pinpointing**: Each candidate answer sentence parse tree is matched against the parse of the question; sometimes also the preceding sentence. As a fallback the window method is used.
- **Ranking of answers**: The candidate answers' scores are compared and the winner(s) are output.

As described later, Webclopedia classifies desired answers by their semantic type, using the approx. 140 classes called *Qtargets* [8,11], which represent the essential point of a question/answer, regardless of phrasing.

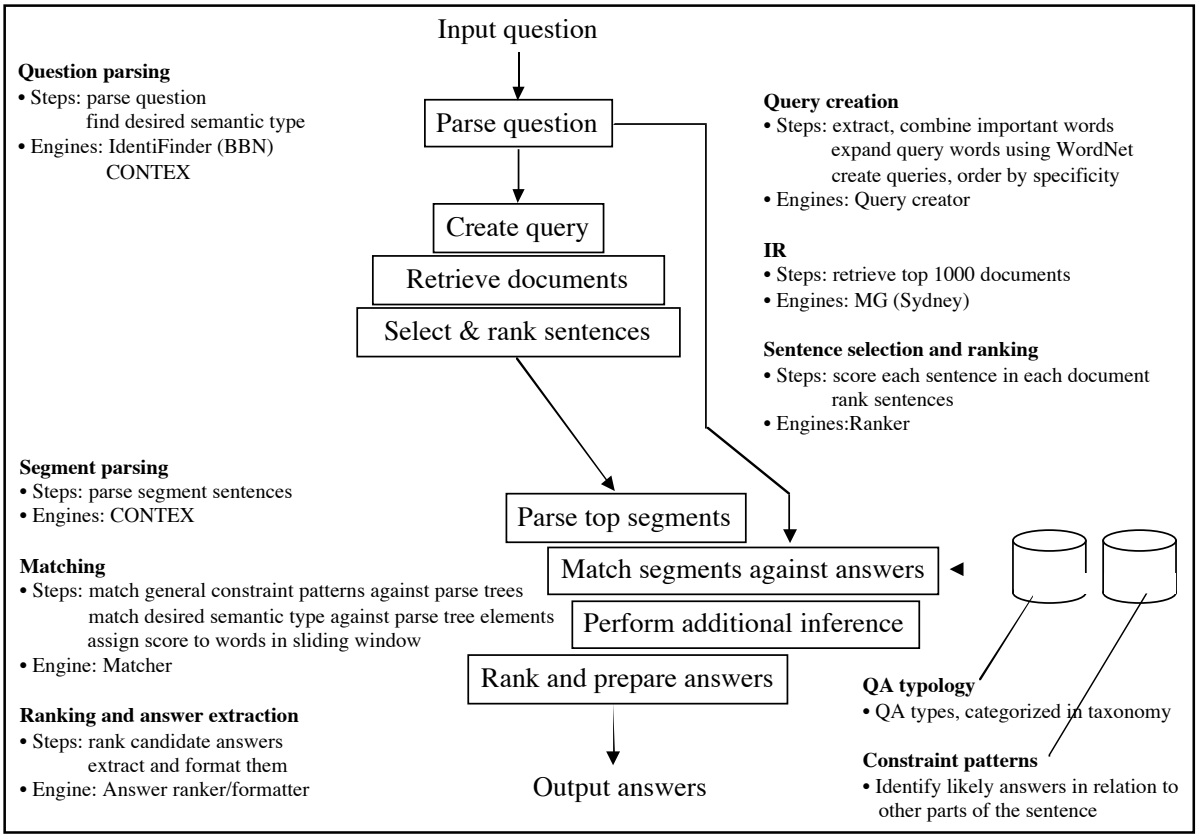

Figure 1. Weclopedia architecture.

3. THE QA TYPOLOGY

Almost all QA systems employ some categorization of answers to help delimit the pinpointing step. The simplest categorization is simple by question word (who, what, when, where, why, and how). More types simplify matching, however. For example, "Who is the richest person in the world?", "Where is Cambodia?", and "When did Marilyn Monroe marry Arthur Miller?" may have answer types *ProperPerson*, *Location*, and *Date* respectively.

The same answer type can be intended by various forms of question, as in "What is the name of the person who invented xeroxing?", "Who invented xeroxing?", and "Who was the inventor of xeroxing?". Similarly, the answer can occur in different forms, such as "Chester F. Carlson invented xeroxing", "the man who invented photocopying was Chester Carlson", and "Inventing the xeroxing process was a high point of Chester F. Carlson's life". What we called an answer type is thus a kind of equivalence class of all these phrasings, or a relation that links all the question forms to all their answer forms. We call this equivalence class a *Qtarget*; for this example it might be *Person*. Most QA systems include a list of such Qtargets, typically ranging in number from around 10 to around 50.

Since many QA systems associate specific matching information (indicative words, surface word patterns, etc.) with their Qtargets, it is useful to create more specific alternatives that narrow the equivalent sets. Thus *Person* might be specialized to *Inventor*, and be associated with words such as "invent", "invention", "discover", "discovery", and "create". Other specializations of *Person* might be *Artist* (with "perform", "sing") and *Author* ("write", "book", "publish"). In addition, questions often explicitly require more or less specific answers ("Where is Vesuvius?" vs. "In which country is Vesuvius?")

The hierarchicalized Qtargets form a typology that defines the types of questions the system can handle. The hierarchicalization can be exploited for backoff matches, to allow more general Qtargets to apply in cases where specific ones fail. QA lists or typologies are reported in almost all QA system papers; see for example [5,1].

ISI'S QA Typology

Over the past two years, we have created at ISI a QA Typology that currently contains 140 Qtargets. Our initial Typology of about 75 nodes was derived from an analysis by one of our students of over 17,000 questions, downloaded from answers.com [8,11]; see http://www.isi.edu/natural-language/projects/webclopedia/Taxonomy/taxonomy_toplevel.html and the description in [11]. Subsequently, we have been restructuring and extending the Typology to its current form.

Qtargets are of several types.

Abstract Qtargets. Some Qtargets are not typical semantic types but are specific to QA. For example, the usual reading of "who was Mother Theresa?" is "why is the individual known as Mother Theresa famous?". The Qtarget A-WHY-FAMOUS is not a semantic class. Additional Qtargets include A-YES-NO-QUESTION and A-ABBREVIATION-EXPANSION.

Semantic (concept) Qtargets. These Qtargets, the largest class, limit the search space to sentence constituents that satisfy a particular semantic class with respect to the Webclopdia ontology, which currently contains about 10,000 items, mostly extracted from WordNet. Semantic Qtargets include C-PROPER-ORGANIZATION, C-BODY-PART C-COLOR, C-PROPER-ANIMAL.

Syntactic Qtargets. Other Qtargets apply when the system cannot determine a specific semantic type for the answer, but can specify the desired syntactic type. Syntactic Qtargets are fairly weak, in that they generally don't restrict the search space much. Webclopedia uses *S-NP* as the default Qtarget. The four syntactic Qtargets are:

S-NP, S-NOUN
What does Peugeot manufacture?
S-VP
What did John Hinckley do to impress Jodie Foster?
S-PROPER-NAME

Role Qtargets. These Qtargets specify constituents of the parse tree of the question and candidate answer:

ROLE REASON
Why did David Koresh ask the FBI for a word processor?
ROLE MANNER
How did David Koresh die?

For example, in the (simplified) parse tree

The tournament was cancelled due to bad weather.
((SUBJ LOG-OBJ) The tournament
(PRED) was cancelled
(REASON) due to bad weather
(DUMMY) .
)

the phrase "due to the bad weather" satisfies the Qtarget ROLE REASON. This constraint is independent of the syntactic category, which also could have been a subordinate clause ("because the weather was so bad") or a verb phrase ("to avoid injuries").

Slot Qtargets. Slot Qtargets refer to non-syntactic information associated with sentence constituents. Slots may be filled during parsing or later. Some examples are:

SLOT TITLE-P TRUE
Name a novel written by Proust.
SLOT QUOTE-P TRUE
What did Richard Feynman say upon hearing he would receive the Nobel Prize in Physics?
SLOT POSSIBLE-REASON-P TRUE

ISI's QA systems Webclopedia [11] and Textmap [12] both employ the Typology. Both systems can combine Qtargets, using variable strengths:

Question: Where is the Getty Museum?
Qtarget: ((C-PROPER-CITY 1.0)
(C-AT-LOCATION 0.7)
(C-PROPER-PLACE 0.7) …)

indicating that the system would prefer a proper city, but could accept something tagged by the named entity tagger just as a general location, or as a place with a name [8].

Relation Qtargets. Relation Qtargets express some relationship between two semantic types, such as *Person* and *Date* to express the Qtarget R-BIRTHYEAR or *Person* and *Noun* to express R-INVENTION.

To find instances of Qtargets in sentences, Webclopedia includes several dozen hand-built matching rules. We have developed a method to learn additional ones automatically. This method was suggested by several recent attempts at a surface-oriented pattern matching approach to QA. The Insight system from Moscow [15] used some hundreds of surface-level patterns to identify answer strings without (apparently) applying Qtargets or similar reasoning. For example, for *BirthYear* questions such as "which year was Mozart born?" the phrase "Mozart (1756 – 1791)…" provides the answer using the general template

NAME_OF_PERSON (BIRTHYEAR – DEATHYEAR)

Several other systems also defined word-level patterns indicating specific Qtargets; e.g., [13]. The Microsoft system [3] extended the idea of a pattern to its limit, by reformulating the input question as a declarative sentence and then retrieving the sentence verbatim, with its answer as a completion, from the web using the normal search engines. For example, "who was Chester F. Carlson?" would be transformed to "Chester F. Carlson was" and submitted. Although this approach might yield many wrong answers (including "Chester F. Carlson was born February 8, 1906, in Seattle"), the sheer number of correct answers often leads to success.

We estimate that word-level patterns can provide at least 25% of the MRR score defined for NIST's TREC QA evaluation competition (although some systems claimed considerably higher results; see [15]). In order to determine their power and reap their benefits, we collected all the patterns associated with as many Qtargets as made sense (some Qtargets, such as C-PROPER-PLANET and C-PROPER-OCEAN, are known closed sets that require no patterns).

As described in [11] we developed an automated procedure to learn such patterns from the web, using Altavista (because it returns 1000 documents per query), and to measure their Precision. More formally this experiment can be phrased as "Given a QA pair such as (NAME_OF_PERSON BIRTHYEAR), extract from the web all the different patterns (TEMPLATEs) that contain this QA pair along with the precision of each pattern". We have inserted into the Typology the patterns for approx. 20 Qtargets, recording their Precision scores and relative frequencies of appearance.

The results were quite good in some cases. For the rather straightforward R-BIRTHYEAR patterns are:

Precision	#Correct	# Found	Pattern
1.0	122	122	<NAME> (<BD> - <DD>
1.0	15	15	<NAME> (<BD> - <DD>) ,
1.0	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>

Note the overlaps among patterns. By not compressing them further we can record different precision levels. Other patterns are provided in the Typology and in [11].

4. THE PROCEDURE

Parsing Questions

Webclopedia parses input questions using CONTEX to obtain a semantic representation of the questions. CONTEX is a deterministic machine-learning based grammar learner/parser that was originally built for MT [6]. For English, parses of unseen sentences measured 87.6% labeled precision and 88.4% labeled recall, trained on 2048 sentences from the Penn Treebank. Over the past few years CONTEX has been extended to Japanese and Korean [7].

For Webclopedia, the CONTEX grammar had to be augmented to recognize Qtargets [8]. For example, given the question "How far is it from Denver to Aspen?", the parser identifies noun phrases, nouns, verb phrases, verbs, adjective phrases, and adjectives embedded in the question, and determines that the question is asking for a distance quantity.

IR: Creating and Expanding Queries

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 [2], which isolates proper names in a text and classifies them as person, organization, or location.

Query expansion comes from two sources and is used in different stages. In the document retrieval stage, the highly relevant question terms (identified by CONTEX) are expanded in order to boost recall, for example going from "Russian" to "Soviet" or from "capital of the United States" to "Washington". In the sentence ranking stage, we use WordNet 1.6 [4] to match expanded query terms. Although these expanded terms contribute to the final score, their contribution is discounted. This application of expansion strategy aims to achieve high precision and moderate recall.

Retrieving Documents

We use MG [16] as our search engine. Although MG is capable of performing ranked query, we only use its Boolean query capability. For the entire TREC-10 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. We order queries most specific first, then gradually relax them to more general, until we have retrieved a sufficient number of documents. For example, *(Denver&Aspen)* is sent to the database first. If the number of documents returned is less than a pre-specified threshold, for example, 500, then we retain this set of documents as the basis for further processing, while also submitting the separate queries *(Denver)* and *(Aspen)*.

Ranking Sentences

If the total numbers of sentences contained in the documents returned by MG is N for a given Boolean query, we would like to rank the sentences in the documents to maximize answer recall and precision in the topmost $K << N$, in order to minimize the parsing and subsequent processing. In this stage we set K=300. We assign goodness score to a sentence according to the following criteria:

1. Exact match of proper names such as "Denver" and "Aspen" get 100% bonus score.
2. Upper case term match of length greater than 1 get 60% bonus, otherwise get 30%. For example, match of "United States" is better than just of "United".
3. Lower case matches get the original score.
4. Lower case term match with WordNet expansion stems get 10% discount. If the original term is capital case then it gets 50% discount. For example, when *Cag(e)* matches *cag(e)*, the former may be the last name of some person while the latter is an object; therefore, the case mismatch signals less reliable information.
5. Lower case term matches after Porter stemming get 30% discount. If the original term is capital case then 70% discount. The Porter stemmed match is considered less reliable than a WordNet stem match.
6. Porter stemmer matches of both question word and sentence word get 60% discount. If the original term is capital case then get 80% discount.
7. If CONTEX indicates a term as being *qsubsumed* then it gets 90% discount. For example, "Which country manufactures weapons of mass destruction?" where "country" will be marked as qsubsumed.

Normally common words are ignored unless they are part of a phrase in question word order. Based on these scores, the total score for a sentence is:

Sentence score = sum of word scores

At the end of the ranking we apply Qtarget filtering to promote promising answer sentences. For example, since the question "How far is it from Denver to Aspen?" is asking for a distance quantity, any sentence that contains only "Denver" or "Aspen" but not any distance quantities are thrown out. Only the top 300 remaining sentences are passed to the answer pinpointing module.

The bonus and discount rates given here are heuristics. We are in the process of developing mechanisms to learn these parameters automatically.

Answer Matching using Qtarget-Specific Knowledge

Once the candidate answer passages have been identified, their sentences are parsed by CONTEX.

The Matcher module then compares their parse trees to the parse tree of the original question. The Matcher performs two independent matches [8,9]:

- match Qtargets and Qargs/Qwords obtained from the questions to those identified in the candidate answer parse trees,
- if that fails, move a word window over the answer text, assigning a score to each position based on word overlaps, etc. [9].

Qtargets and their accompanying Qargs enable the matcher to pinpoint within the answer passage the exact, syntactically delimited, answer segment. (In contrast, word window matching techniques, that have no recourse to parse structures, have no accurate way to delimit the exact answer boundaries.)

For many questions, the Qtarget, syntactic clues, and word overlap are insufficient to select a good answer. Strategies for dealing with this problem are described in [9,10].

In NIST's TREC QA evaluation competition, Webclopedia tied for second place in 2000 and scored fourth in a field of over 30 in 2001.

5. CONCLUSION

Ongoing work [12] is focusing on answer longer, more complex questions, including ones with somewhat structured answers (such as biographies and descriptions of events and objects) and ones expressing opinions.

6. REFERENCES

[1] Abney, S., M. Collins, and A. Singhal. 2000. Answer Extraction. *Proceedings of the Applied Natural Languge Processing Conference (ANLP)*. Seattle, WA (296–301).

[2] 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.

[3] Brill, E., J. Lin, M. Banko, S. Dumais, and A. Ng. 2001. Data-Intensive Question Answering. *Proceedings of the TREC-10 Conference*. NIST, Gaithersburg, MD (183–189).

[4] Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. Cambridge: MIT Press.

[5] Harabagiu, S., M. Pasca, and S. Maiorano. 2000. Experiments with Open Domain Textual Question Answering. *Proceedings of the 18th COLING Conference.* Saarbrücken, Germany (292–298).

[6] 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.

[7] 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.

[8] Hermjakob, U. 2001. Parsing and Question Classification for Question Answering. In *Proceedings of the Workshop on Question Answering at the Conference ACL-2001*. Toulouse, France.

[9] Hovy, E.H., L. Gerber, U. Hermjakob, M. Junk, and C.-Y. Lin. 2000. Question Answering in Webclopedia. *Proceedings of the TREC-9 Conference*. NIST. Gaithersburg, MD.

[10] Hovy, E.H., U. Hermjakob, and C.-Y. Lin. 2001. The Use of External Knowledge in Factoid QA. *Proceedings of the TREC-10 Conference*. NIST, Gaithersburg, MD (166–174).

[11] Hovy, E.H., U. Hermjakob, C.-Y. Lin, and D. Ravichandran. 2002. A Question/Answer Typology with Surface Text Patterns. *Notebook of the Human Language Technology conference (HLT)*. San Diego, CA.

[12] Marcu, D., E.H. Hovy, and K. Knight. 2001. Textmap. Presentation at the ACQUAINT Kickoff Meeting. Washington, DC, December 2001.

[13] Oh, JH., KS. Lee, DS. Chang, CW. Seo, and KS. Choi. 2001. TREC-10 Experiments at KAIST: Batch Filtering and Question Answering. *Proceedings of the TREC-10 Conference*. NIST, Gaithersburg, MD (354–361).

[14] Ravichandran, D. and E.H. Hovy. 2002. *Proceedings of the ACL Conference*. Philadelphia, PA.

[15] Soubbotin, M.M. and S.M. Soubbotin. 2001. Patterns of Potential Answer Expressions as Clues to the Right Answer. *Proceedings of the TREC-10 Conference*. NIST, Gaithersburg, MD. (175–182).

[16] Witten, I.H., A. Moffat, and T.C. Bell. 1994. *Managing Gigabytes: Compressing and Indexing Documents and Images*. New York: Van Nostrand Reinhold.