LASSO: A Tool for Surfing the Answer Net

Dan Moldovan, Sanda Harabagiu,
Marius Paşca, Rada Mihalcea, Richard Goodrum, Roxana Gîrju and **Vasile Rus**
Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275-0122
{moldovan,sanda,mars,rada,goodrum,roxana,rus}@seas.smu.edu

Abstract

This paper presents the architecture, operation and results obtained with the LASSO system developed in the Natural Language Processing Laboratory at SMU. The system relies on a combination of syntactic and semantic techniques, and lightweight abductive inference to find answers. The search for the answer is based on a novel form of indexing called paragraph indexing. A score of 55.5% for short answers and 64.5% for long answers was achieved.

Background

Finding the answer to a question by returning a small fragment of a text, where the answer actually lies, is profoundly different from the task of information retrieval (IR) or information extraction (IE). Current IR systems allow us to locate full documents that might contain the pertinent information, leaving it to the user to extract the answer from a ranked list of texts. In contrast, IE systems extract the information of interest, provided it has been presented in a predefined, target representation, known as *template*. The immediate solution of combining IR and IE techniques for question/answering (Q/A) is impractical, since IE systems are known to be highly dependent on domain knowledge, and furthermore, the generation of templates is not performed automatically.

Our methodology of finding answers in large collections of documents relies on natural language processing (NLP) techniques in novel ways. First, we perform a processing of the question by combining syntactic information, resulting from a shallow parse, with semantic information that characterizes the question (e.g. *question type*, *question focus*). Secondly, the search for the answer is based on a novel form of indexing, called *paragraph indexing* and new related retrieval methods. Finally, in order to extract the answers and to evaluate their correctness, we use a battery of abductive techniques, some based on empirical methods, some on lexico-semantic information. The principles that have guided our paragraph indexing and the abductive inference of the answers are reported in (Harabagiu and Maiorano 1999).

When designing LASSO, the Q/A system developed by the NLP group at SMU, our goal was not to employ NLP techniques just for enhancing the IR results. Instead, we developed a Q/A model that retains the elegance of IR systems, by using shallow processing, and adds the exactness of IE systems, by providing with methods of finding and extracting answers without deep NLP. Furthermore, to comply with the open-domain constraints of the TREC Q/A task, we relied only on lexico-semantic resources that are of general nature. This design allows the escalation to Q/A systems capable of handling questions that impose high-level reasoning techniques (e.g. questions used in the evaluations of the *High Performance Knowledge Bases (HPKB) program* (Cohen et al.1998).

Overview of the LASSO Q/A System

The architecture of LASSO comprises three modules: *Question Processing* module, *Paragraph Indexing* module and *Answer Processing* module. Given a question, of open-ended nature, expressed in natural language, we first process the question by creating a representation of the information requested. Thus we automatically find (a) what *type of question* it is, from the taxonomy of questions at hand, (b) what *type of answer* is expected, and most importantly, (c) what is the *question focus* defined as the main information required by the interrogation. Furthermore, the *Question Processing* also identifies the keywords from the question, which are passed to the *Paragraph Indexing* module, as illustrated by Figure 1.

In LASSO, documents are indexed by a modified Zprise IR system available from NIST. Our search engine incorporates a set of Boolean operators (e.g. AND, OR, NOT, NEAR). We post-process the results of the IR search engine by filtering out the returns that do not contain all keywords in the same paragraph. This op-

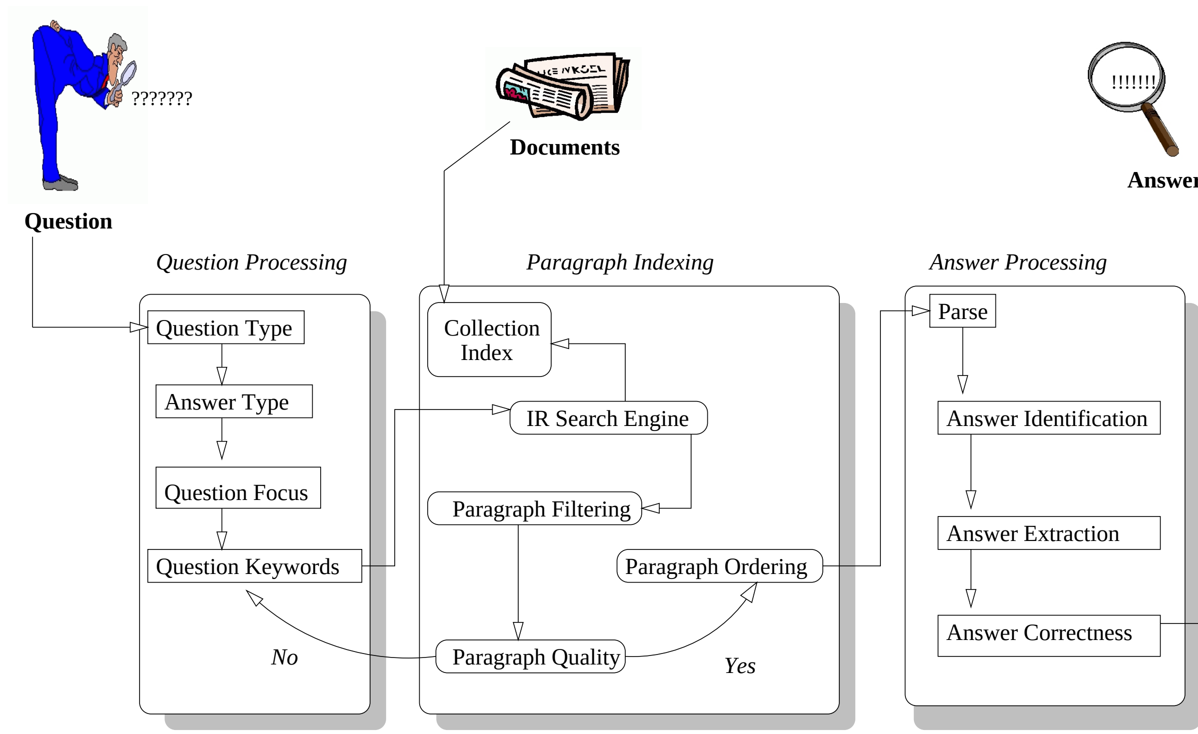

Figure 1: Architecture of the LASSO Q/A System

eration allows for on-the-fly generation of a paragraph index. The second important feature of the *Paragraph Indexing* module comes from the evaluation of the quality of the paragraphs. When the quality is satisfactory, we order the paragraphs according with a plausibility degree of containing the answer. Otherwise, we add/drop keywords and resume the paragraph retrieval. This loop generates a feed-back retrieval context that enables only a reasonable number of paragraphs to be passed to the *Answer Processing* module.

The advantage of processing paragraphs instead of full documents determines a faster syntactic parsing. Our parses also involve Named Entity recognitions and use of lexico-semantic resources that are valuable in the extraction of the answer. The extraction and evaluation of the answer correctness is based on empirical abduction.

Question Processing

The role of the question processing module is to: (1) determine the type of question, (2) determine the type of answer expected, (3) build a focus for the answer, and (4) transform the question into queries for the search engine.

In order to find the right answer to a question from a large collection of texts, first we have to know what we should look for. The answer type can usually be determined from the question. For a better detection of the answer, the questions are first classified by their type: *what*, *why*, *who*, *how*, *where* questions, etc. A further classification follows to better identify the question type. Table 1 shows the classification for the 200 TREC-8 questions. "

We further realized that the question type was not sufficient for finding answers. For the questions like *Who was the first American in space?*, the answer type is obvious: PERSON. However, this does not apply for example to the questions of type *what*, as *what* is ambiguous and it says nothing about the information asked by the question. The same applies to many other question types. The problem was solved by defining a concept named *focus*.

A *focus* is a word or a sequence of words which define the question and disambiguate it in the sense that it indicates what the question is looking for, or what the question is all about. For example, for the question *What is the largest city in Germany?*, the focus is *largest city*. Knowing the focus and the question type it becomes easier to determine the type of the answer sought, namely: the name of the largest city in Germany.

The focus is also important in determining the list of keywords for query formation. We noticed that some

Q-class	Q-subclass	Nr. Q	Nr. Q answered	Answer type	Example of question	Focus
what		64	54			
	basic what	40	34	MONEY/NUMBER/ DEFINITION/TITLE/ NNP/UNDEFINED	*What was the monetary value of the Nobel Peace Prize in 1989?*	monetary value
	what-who	7	7	PERSON/ ORGANIZATION	*What costume designer decided that Michael Jackson should only wear one glove?*	costume designer
	what-when	3	2	DATE	*In what year did Ireland elect its first woman president?*	year
	what-where	14	12	LOCATION	*What is the capital of Uruguay?*	capital
who		47	37	PERSON/ ORGANIZATION	*Who is the author of the book "The Iron Lady: A Biography of Margaret Thatcher"?*	author
how		31	21			
	basic how	1	0	MANER	*How did Socrates die?*	Socrates
	how-many	18	13	NUMBER	*How many people died when the Estonia sank in 1994?*	people
	how-long	2	2	TIME/DISTANCE	*How long does it take to travel from Tokyo to Niigata?*	–
	how-much	3	2	MONEY/PRICE	*How much did Mercury spend on advertising in 1993?*	Mercury
	how-much-<modifier>	1	0	UNDEFINED	*How much stronger is the new vitreous carbon material invented by the Tokyo Institute of Technology compared with the material made from cellulose?*	new vitreous carbon material
	how-far	1	1	DISTANCE	*How far is Yaroslavl from Moscow?*	Yaroslavl
	how-tall	3	3	NUMBER	*How tall is Mt. Everest?*	Mt. Everest
	how-rich	1	0	UNDEFINED	*How rich is Bill Gates?*	Bill Gates
	how-large	1	0	NUMBER	*How large is the Arctic refuge to preserve unique wildlife and wilderness value on Alaska's north coast?*	Arctic refuge
where		22	16	LOCATION	*Where is Taj Mahal?*	Taj Mahal
when		19	13	DATE	*When did the Jurassic Period end?*	Jurassic Period
which		10	8			
	which-who	1	1	PERSON	*Which former Klu Klux Klan member won an elected office in the U.S.?*	former Klu Klux Klan member
	which-where	4	3	LOCATION	*Which city has the oldest relationship as sister-city with Los Angeles?*	city
	which-when	1	1	DATE	*In which year was New Zealand excluded from the ANZUS alliance?*	year
	which-what	4	3	NNP/ ORGANIZATION	*Which Japanese car maker had its biggest percentage of sale in the domestic market?*	Japanese car maker
name		4	4			
	name-who	2	2	PERSON/ ORGANIZATION	*Name the designer of the show that spawned millions of plastic imitations, known as "jellies"?*	designer
	name-where	1	1	LOCATION	*Name a country that is developing a magnetic levitation railway system?*	country
	name-what	1	1	TITLE/NNP	*Name a film that has won the Golden Bear in the Berlin Film Festival?*	film
why		2	0	REASON	*Why did David Koresh ask for a word processor?*	David Koresh
whom		1	0	PERSON/ ORGANIZATION	*Whom did the Chicago Bulls beat in the 1993 championship?*	Chicago Bulls
Total		200	153 77%			

Table 1: Types of questions and statistics. In this table we considered that a question was answered correctly if its answer was among top five ranked long answers.

words in the questions never occur in the answer, and that is because their role is just to disambiguate the question. For example, in the question *In 1990, what day of the week did Christmas fall on?*, the focus is *day of the week*, a concept that is unlikely to occur in the answer. In such situations, the focus should not be included in the list of keywords considered for detecting the answer.

The process of extracting keywords is based on a set of ordered heuristics. Each heuristic returns a set of keywords, that is added in the same order to the question keywords. We have implemented eight different heuristics. Initially, only the keywords returned by the first six heuristics are considered. If further keywords are needed in the retrieval loop, keywords provided by the other two heuristics are added. When keywords define an exceedingly specific query, they are dropped in the reversed order in which they have been entered. The heuristics are:

- *Keyword-Heuristic 1*: Whenever quoted expressions are recognized in a question, all non-stop words of the quotation became keywords.
- *Keyword-Heuristic 2*: All named entities, recognized as proper nouns, are selected as keywords.
- *Keyword-Heuristic 3*: All complex nominals and their adjectival modifiers are selected as keywords.
- *Keyword-Heuristic 4*: All other complex nominals are selected as keywords.
- *Keyword-Heuristic 5*: All nouns and their adjectival modifiers are selected as keywords.
- *Keyword-Heuristic 6*: All the other nouns recognized in the question are selected as keywords.
- *Keyword-Heuristic 7*: All verbs from the question are selected as keywords.
- *Keyword-Heuristic 8*: The question focus is added to the keywords .

Table 2 lists two questions from the TREC-8 competition together with their associated keywords. The Table also illustrates the trace of keywords until the paragraphs containing the answer were found. For question 26, the paragraphs containing the answers could not be found before dropping many of the initial keywords. In contrast, the answer for question 13 was found when the verb *rent* was added to the Boolean query.

Q-26	*What is the name of the "female" counterpart to El Nino, which results in cooling temperatures and very dry weather ?*
Keys	female El Nino dry weather cooling temperatures female El Nino dry weather cooling female El Nino dry weather female El Nino dry female El Nino female El
Q-13	*How much could you rent a Volkswagen bug for in 1966 ?*
Keys	Volkswagen bug *Volkswagen bug rent*

Table 2: Examples of TREC-8 Question Keywords

Paragraph Indexing

Search engine

The Information Retrieval Engine for LASSO is related to the Zprise IR search engine available from NIST. There were several features of the Zprise IR engine which were not conducive to working within the design of LASSO. Because of this, a new IR engine was generated to support LASSO without the encumbrance of these features. The index creation was, however, kept in its entirety.

The Zprise IR engine was built using a cosine vector space model. This model does not allow for extraction of those documents which include all of the keywords, but extracts documents according to the similarity measure between the document and the query as computed by the cosine of the angle between the vectors represented by the document and the query. This permits documents to be retrieved when only one of the keywords is present. Additionally, the keywords present in one retrieved document may not be present in another retrieved document.

LASSO's requirements are much more rigid. LASSO requires that documents be retrieved only when all of the keywords are present in the document. Thus, it became necessary to implement a more precise determinant for extraction. For the early work, it was determined that a Boolean discriminate would suffice provided that the operators AND and OR were implemented. It was also necessary to provide the ability to organize queries through the use of parentheses.

We opted for the Boolean indexing as opposed to vector indexing because Boolean indexing increases the *recall* at the expense of *precision*. That works well for us since we control the retrieval precision with the PARAGRAPH operator which provides document filtering. In addition, the Boolean indexing requires less processing time than vector indexing, and this becomes important when the collection size increases.

To facilitate the identification of the document sources, the engine was required to put the document id in front of each line in the document.

The index creation includes the following steps: normalize the SGML tags, eliminate extraneous characters, identify the words within each document, stem the terms (words) using the Porter stemming algorithm, calculate the local (document) and global (collection) weights, build a comprehensive dictionary of the collection, and create the inverted index file.

The index generation process for LASSO is the same process as used by Zprise, however, several minor changes were necessary for the inclusion of the data presented.

It was observed that while the Zprise index process should work for multiple databases, it did not. Since there are four distinct sources of data present in the collection, four unique indices were created (one for each source). The fact that LASSO uses a Boolean discriminate versus a cosine vector space similarity measure makes this permissible. Furthermore, it became necessary to expand the number of SGML tags that are included in the index creation process. This was necessary since each source chose to use a different, but overlapping, set of SGML tags.

Paragraph filtering

The number of documents that contain the keywords returned by the Search Engine may be large since only weak Boolean operators were used. A new, more restrictive operator was introduced: PARAGRAPH n. This operator searches like an AND operator for the words in the query with the constraint that the words belong only to some n consecutive paragraphs, where n is a controllable positive integer.

The parameter n selects the number of paragraphs, thus controlling the size of the text retrieved from a document considered relevant. The rationale is that most likely the information requested is found in a few paragraphs rather than being dispersed over an entire document.

In order to apply this new operator, the documents retrieved by the search engine have to be segmented into sentences and paragraphs. Separating a text into sentences proves to be an easy task, one could just make use of the punctuation to solve this problem. However, the paragraph segmentation is much more difficult, and this is due to the highly unstructured texts that can be found in a collection. Thus, we had to use a method that covers almost all the possible paragraph separators that can occur in the texts. The paragraph separators that were implemented so far are: (1) HTML tags, (2) empty lines and (3) paragraph indentations.

Paragraph ordering

Paragraph ordering is performed by a radix sort that involves three different scores: the largest *Same_word_sequence-score*, the largest *Distance-score* and the smallest *Missing_keyword-score*. The definition of these scores is based on the notion of *paragraph-window*. Paragraph-windows are determined by the need to consider separately each match of the same keyword in the same paragraph. For example, if we have a set of keyword {*k1, k2, k3, k4*} and in a paragraph *k1* and *k2* are matched each twice, whereas *k3* is matched only once, and *k4* is not matched, we are going to have four different windows, defined by the keywords: [*k1-match1, k2-match1, k3*], [*k1-match2,k2-match1, k3*], [*k1-match1, k2-match2, k3*], and [*k1-match2, k2-match2, k3*]. A window comprises all the text between the lowest positioned keyword in the window and the highest position keyword in the window. Figure 2 illustrates the four windows for our example.

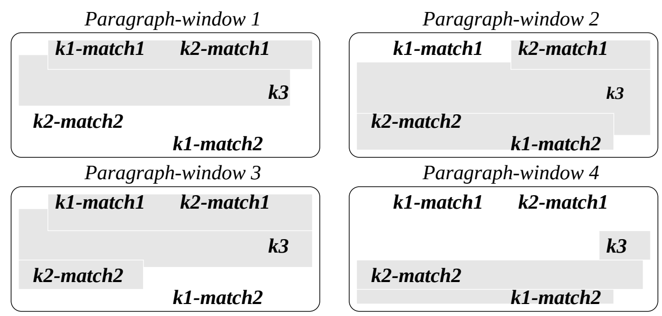

Figure 2: Four windows defined on the same paragraph

For each paragraph window we compute the following scores:

• *Same_word_sequence-score*: computes the number of words from the question that are recognized in the same sequence in the current paragraph-window.

• *Distance-score*: represents the number of words that separate the most distant keywords in the window.

• *Missing_keywords-score*: computes the number of unmatched keywords. This measure is identical for all windows from the same paragraph, but varies for windows from different paragraphs.

The radix sorting takes place across all the window scores for all paragraphs.

Answer Processing

The *Answer Processing* module identifies and extracts the answer from the paragraphs that contain the question keywords. Crucial to the identification of the answer is the recognition of the answer type. Since almost always the answer type is not explicit in the question, nor in the answer, we need to rely on lexico-semantic information, provided by a parser that identifies named entities (e.g. names of people or organizations), monetary units, dates and temporal/locative expressions, as well as products. The recognition of the answer type, through the semantic tag returned by the parser, creates a *candidate answer*. The extraction of the answer and its evaluation are based on a set of heuristics.

The Parser

The parser combines information from broad coverage lexical dictionaries with semantic information that contributes to the identification of the named entities. Since part-of-speech tagging is an intrinsic component of a parser, we have extended Brill's part-of-speech tagger in two ways. First, we have acquired new tagging rules and secondly, we have unified the dictionaries of the tagger with semantic dictionaries derived from the Gazetteers and from WordNet (Miller 1995). In addition to the implementation of grammar rules, we have implemented heuristics capable of recognizing names of persons, organizations, locations, dates, currencies and products. Similar heuristics recognize named entities successfully in IE systems. Having these capabilities proved to be useful for locating the possible answers within a set of candidate paragraphs.

Answer Extraction

The parser enables the recognition of the *answer candidates* in the paragraph. Each expression tagged by the parser with the answer type becomes one of the answer candidates for a paragraph. Similar to the paragraph-windows used in ordering the paragraphs, we establish an *answer-window* for each answer candidate. To evaluate the correctness of each answer candidate, a new evaluation is computed for each answer-window. We use the following scores:

- *Same_word_sequence-score*: it is computed in the same way as for *paragraph-windows*.
- *Punctuation_sign-score*: is a flag set when the answer candidate is immediately followed by a punctuation sign.
- *Comma_3_words-score*: measures the number of question words the follow the answer candidate, when the latter is succeeded by a comma. A maximum of three words are sought.
- *Same_parse_subtree-score*: computes the number of question words found in the same parse sub-tree as the answer candidate.
- *Same_sentence-score*: computes the number of question words found in the same sentence as the answer candidate.
- *Matched_keywords-score*: computes the number of keywords matched in the answer-window.
- *Distance-score*: adds the distances (measured in number of words) between the answer candidate and the other question words in the same window.

The overall score for a given answer candidate is computed by:

$$\begin{aligned}\textit{Combined-score} = {} & 16 * \textit{Same\_word\_sequence-score} + \\ & + 16 * \textit{Punctuation\_sign-score} + \\ & + 32 * \textit{Comma\_3\_words-score} + \\ & + 16 * \textit{Same\_parse\_subtree-score} + \\ & + 16 * \textit{Same\_sentence-score} + \\ & + 16 * \textit{Matched\_keywords-score} - \\ & - 4 * \sqrt{Distance - score}\end{aligned}$$

Currently the combined score represents an unnormalized measure of answer correctness. The answer extraction is performed by choosing the answer candidate with the highest score. Some of the scores approximate very simple abductions. For example, the recognition of keywords or other question words in an apposition determines the *Punctuation_sign-score*, the *Same_parse_subtree-score*, the *Comma_3_words-score* and the *Same_sentence-score* to go up. Moreover, the same sequence score gives higher plausibility to answer candidates that contain in their window sequences of question words that follow the same orders in the question. This score approximates the assumption that concepts are lexicalized in the same manner in the question and in the answer. However, the combined score allows for keywords and question words to be matched in the same order.

Table 3 illustrates some of the scores that were attributed to the candidate answers LASSO has extracted successfully. Currently we compute the same score for both short and long answers, as we analyze in the same way the answer windows.

Question-8	What is the name of the rare neurological disease with symptoms such as : involuntary movements (tics), swearing, and incoherent vocalizations (grunts, shouts, etc)?
Answer (short)	Score: 284.40 *who said she has both Tourette's Syndrome and*
Question-34	Where is the actress Marion Davies, buried ?
Answer (short)	Score: 142.56 *from the fountain inside Hollywood Cemetery*
Question-73	Where is the Taj Mahal ?
Answer (long)	Score: 408.00 *list of more than 360 cities throughout the world includes the Great Reef in Australia, the Taj Mahal in India, Chartre's Cathedral in France, and Seregenti National Park in Tanzania. The four sites Japan has listed include*
Question-176	What is the nationality of Pope John Paul II ?
Answer (long)	Score: 407.06 *stabilize the country with its help, the Catholic hierarchy stoutly held out for pluralism, in large part at the urging of Polish-born Pope John Paul II. When the Pope emphatically defended the Solidarity trade union during a 1987 tour of the*

Table 3: Examples of LASSO's correctness scores.

Performance evaluation

Table 4 summarizes the scores provided by NIST for our system.

	Percentage of questions in top 5	NIST score
Short answer	68.1%	55.5%
Long answer	77.7%	64.5%

Table 4: Accuracy performance

Another important performance parameter is the *processing time* to answer a question. On the average, the processing time per question is 6 sec., and the time ranges from 1 sec. to 540 sec. There are four main components of the overall time: (1) paragraph search time, (2) paragraph ordering time, (3) answer extraction time, and (4) question processing time. Compared with the rest, the question processing time is negligible. Figure 3 shows the relative percentage (represented on the vertical axis) of the first three components, for the entire range of overall question processing time. The horizontal axis ranks the questions according with their processing time, from the shortest time of 1 sec. to 540 sec.

It can be seen that as the overall time increases, the answer extraction time (including parsing) tends to represent a higher percentage then the rest, meaning that answer extraction is the module where most of the time is spent. This is important as it indicates a bottleneck in the time performance of the system.

Lessons learned

In principle, the problem of finding one or more answers to a question from a very large set of documents can be addressed by creating a context for the question and a knowledge representation of each document and then match the question context against each document representations. This approach is not practical yet since involves advanced techniques in knowledge representation of open text, reasoning, natural language processing, and indexing that currently are beyond the technology state of the art. On the other hand, traditional information retrieval and extraction techniques alone can not be used for question answering due to the need to pinpoint exactly an answer in large collections of open domain texts. Thus, a mixture of natural language processing and information retrieval methods may be the solution for now.

In order to better understand the nature of the QA task and put this into perspective, we offer in Table 5 a taxonomy of question answering systems. It is not sufficient to classify only the types of questions alone, since for the same question the answer may be easier or more difficult to extract depending on how the answer is phrased in the text. Thus we classify the QA systems, not the questions. We provide a taxonomy based on three criteria that we consider important for building question answering systems: (1) knowledge base, (2) reasoning, and (3) natural language processing and indexing techniques. Knowledge bases and reasoning provide the medium for building question contexts and matching them against text documents. Indexing identifies the text passages where answers may lie, and natural language processing provides a framework for answer extraction.

Out of the 153 questions that our system has answered, 136 belong to Class 1, and 17 to Class 2. Obviously, the questions in Class 2 are more difficult as they require more powerful natural language and reasoning techniques.

As we look for the future, in order to address questions of higher classes we need to handle real-time knowledge acquisition and classification from different domains, coreference, metonymy, special-purpose reasoning, semantic indexing and other advanced techniques.

References

Sergey Brin and Lawrence Page. The anatomy of a Large-Scale Hypertextual Web Search Engine. In the *Proceedings of the Seventh International World Wide Web Conference*, 1998.

Chris Buckley, Mandar Mitra, Janet Walz and Claire Cardie. SMART Hight Precision: TREC 7. In the *Proceedings of the Text Retrieval Conference TREC-7*, 1998.

Paul Cohen, Robert Schrag, Eric Jones, Adam Pease, Albert Lin, Barbara Starr, David Gunning and Murray Burke. The DARPA High Performance Knowledge Bases Project. In *AI Magazine*, Vol 18, No 4, pages 25–49, 1998.

Paul Cohen, Vinay Chaudhri, Adam Pease and Robert Schrag. Does Prior Knowledge Facilitate the Development of Knowledge-Based Systems? In the *Proceedings of AAAI-99*, 1999.

Sanda Harabagiu and Dan Moldovan. A Parallel System for Text Inference Using Marker Propagations. *IEEE Transactions in Parallel and Distributed Systems*, Vol 9, no 8, pages 729–748, 1998.

Sanda Harabagiu and Dan Moldovan. Knowledge Processing on Extended WordNet. In *WordNet: An Electronic Lexical Database and Some of its Applications*, editor Fellbaum, C., MIT Press, Cambridge, MA, 1998.

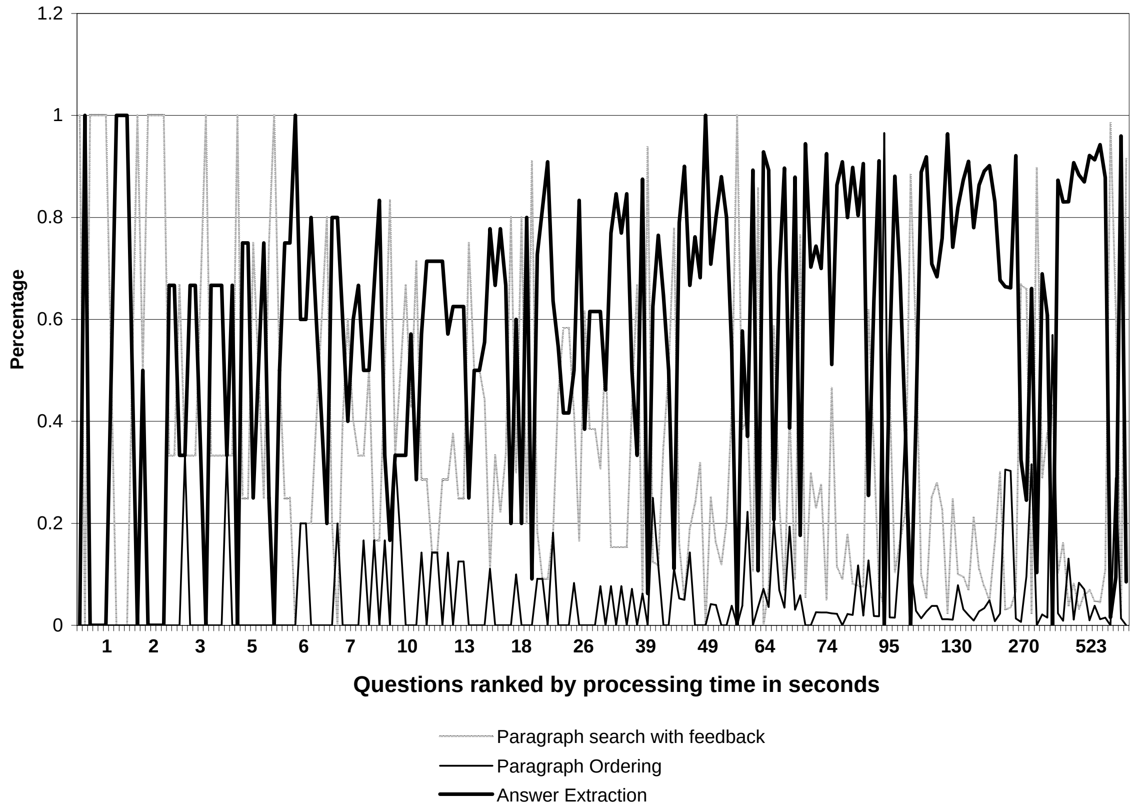

Figure 3: Performance Analysis

Sanda Harabagiu and Steven Maiorano. Finding answers in large collections of texts: paragraph indexing + abductive inference. *Working Notes of the Fall AAAI Symposium on Question Answering*, November 1999.

Marti Hearst. Automated Discovery of WordNet Relations. In *WordNet: An Electronic Lexical Database and Some of its Applications*, editor Fellbaum, C., MIT Press, Cambridge, MA, 1998.

Lynette Hirschman, Marc Light, Eric Breck and John D. Burger. Deep Read: A Reading Comprehension System. In the *Proceedings of the 37th Meeting of the Association for Computational Linguistics (ACL-99)*, pages 325–332, University of Maryland, 1999.

Jerry Hobbs, Mark Stickel, Doug Appelt, and Paul Martin. Interpretation as abduction. *Artificial Intelligence*, 63, pages 69–142, 1993.

Boris Katz. From Sentence Processing to Information Access on the World Wide Web. *Proceedings of the AAAI Spring Symposium*, pages 77–86, 1997.

Julian Kupiec. MURAX: A Robust Linguistic Approach for Question Answering Using an On-line Encyclopedia. In the *Proceedings of the 16th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-93)*, pages 181–190, Pittsburg, PA, 1993.

G.A. Miller. WordNet: A Lexical Database. *Communication of the ACM*, vol 38: No11, pages 39–41, November 1995.

Dan Moldovan and Rada Mihalcea. A WordNet-based Interface to Internet Search Engines. In *Proceedings of the FLAIRS-98*, pages 275–279, 1998.

Gerard A. Salton. Automatic Text Processing: The transformation, analysis and retrieval of information by computer. Addison-Wesley, 1989.

William A. Woods. *Conceptual Indexing: A Better*

Class	KB	Reasoning	NLP/Indexing	Examples and Comments
1	dictionaries	simple heuristics, pattern matching	complex noun, apposition, simple semantics, keyword indexing	Q33: *What is the largest city in Germany?* A: .. *Berlin, the largest city in Germany..* Answer is: simple datum or list of items found verbatim in a sentence or paragraph.
2	ontologies	low level	verb nominalization, semantics, coherence, discourse	Q198: *How did Socrates die?* A: .. *Socrates poisoned himself..* Answer is contained in multiple sentences, scattered throughout a document.
3	very large KB	medium level	advanced nlp, semantic indexing	Q: *What are the arguments for and against prayer in school?* Answer across several texts.
4	Domain KA and Classification, HPKB	high level		Q: *Should Fed raise interest rates at their next meeting?* Answer across large number of documents, domain specific knowledge acquired automatically.
5	World Knowledge	very high level, special purpose		Q: *What should be the US foreign policy in the Balkans now?* Answer is a solution to a complex, possible developing scenario.

Table 5: A taxonomy of Question Answering Systems. The degree of complexity increases from Class 1 to Class 5, and it is assumed that the features of a lower class are also available at a higher class.

way to Organize Knowledge. Technical Report of Sun Microsystems Inc., 1997.