Performance Issues and Error Analysis in an Open-Domain Question Answering System

DAN MOLDOVAN, MARIUS PAŞCA, SANDA HARABAGIU, and MIHAI SURDEANU
Language Computer Corporation

This paper presents an in-depth analysis of a state-of-the-art Question Answering system. Several scenarios are examined: (1) the performance of each module in a serial baseline system, (2) the impact of feedbacks and the insertion of a logic prover, and (3) the impact of various retrieval strategies and lexical resources. The main conclusion is that the overall performance depends on the depth of natural language processing resources and the tools used for answer finding.

Categories and Subject Descriptors: H.5.2 [**Information Interfaces and Presentation**]: User interfaces—*Natural language*; H.3.4 [**Information Storage and Retrieval**]: Systems and software—*Question-answering (fact retrieval) systems*; I.2.1 [**Artificial Intelligence**]: Applications and expert systems—*Natural language interfaces*; I.2.7 [**Artificial Intelligence**]: Natural language processing—*Text analysis*

General Terms: Performance, Experimentation

Additional Key Words and Phrases: Question answering, performance analysis, text retrieval, natural language applications

1. INTRODUCTION

Aiming at returning brief answers in response to natural language questions, open-domain Question Answering (QA) systems represent an advanced application of natural language processing. The global metrics used in the QA track evaluations of the Text REtrieval Conference (TREC) [Voorhees 1999] allow for the overall assessment of the QA system performance. As part of a relentless quest to improve QA systems, it is necessary to measure not only the global performance, but also the performance of each individual module and other architectural features. A detailed performance analysis indicates not only that the system fails to provide an answer but why the system failed. The performance analysis is useful to system designers who want to identify error sources and weak modules.

The QA literature from the last few years reports on global performance of various systems [Abney et al. 2000; Hovy et al. 2001]. It was shown that global

Authors' addresses: 1701 North Collins Blvd., Suite 2200, Richardson, TX 75080; email: {moldovan, marius,sanda,mihai}@languagecomputer.com.

performance depends on factors such as the answer redundancy, the collection size [Clarke et al. 2001, 2002; Breck et al. 2001], and others. General evaluation metrics have been discussed in Voorhees and Tice [2000] and Breck et al. [2000]. However, with few exceptions [Ittycheriah et al. 2001; Light et al. 2001], little has been said about in-depth error analysis in QA systems.

Since most QA systems consist of modules that are chained serially [Abney et al. 2000; Prager et al. 2000], the overall performance is controlled by their weakest link. In this case the error analysis is straightforward. Our system architecture uses several feedbacks, which complicates significantly the error analysis.

This paper presents an in-depth performance analysis of a state-of-the-art QA system. Several configurations are examined: first, the performance of each module in a baseline chained architecture, then, the impact of feedbacks and the insertion of new advanced modules, and finally, the impact of various lexical resources. Our QA system was ranked high in the last three TREC QA track evaluations (cf. Voorhees [1999]). The system is organized around three higher-level subtasks, namely, question processing, document and passage retrieval, and answer processing. Therefore the results are representative of other QA serial architectures that internally perform equivalent subtasks [Abney et al. 2000; Cardie et al. 2000; Ittycheriah et al. 2001] or employ similar lexical resources and tools [Hovy et al. 2001; Prager et al. 2000].

2. TAXONOMY

The performance of a QA system is tightly coupled with the complexity of questions asked and the difficulty of answer extraction. For example, in TREC many systems were quite successful at providing correct answers to simpler, fact-seeking questions, but failed to answer questions that required reasoning or advanced linguistic analysis [Voorhees 1999]. Thus 70% of the participating systems returned a correct answer to question Q1013: *"Where is Perth?"* On the other hand, none could find a correct answer to complex questions such as Q1165: *"What is the difference between AM radio stations and FM radio stations?"*

Since performance is affected by the complexity of question processing, we first provide a broad taxonomy of systems.

2.1 Criteria

The taxonomy is based on several criteria that play an important role in building QA systems: (a) linguistic and knowledge resources, (b) natural language processing involved, (c) document processing, (d) reasoning methods, (e) whether system assumed answers are explicitly stated in a document, (f) whether answer fusion is necessary.

2.2 Classes of QA Systems

Class 1. QA systems capable of processing factual questions. These systems extract answers as text snippets from one or more documents. Often the answer is found verbatim in a text or as a simple morphological variation.

Typically the answers are extracted using empirical methods relying on keyword manipulations.

Class 2. *QA systems enabling simple reasoning mechanisms.* The characteristic of this class is that answers are found in snippets of text, but unlike in Class 1, inference is necessary to relate the question with the answer. More elaborate answer detection methods such as ontologies or codification of pragmatic knowledge are necessary. Semantic alternations, world knowledge axioms, and simple reasoning methods are necessary. An example is Q198—"*How did Socrates die?*"—where *die* has to be linked with *drinking poisoned wine*. WordNet and its extensions are sometimes used as sources of world knowledge.

Class 3. *QA systems capable of answer fusion from different documents.* In this class, the partial answer information is scattered throughout several documents and answer fusion is necessary. The complexity here ranges from assembling simple lists to far more complex questions like script questions, (e.g., "*How do I assemble a bicycle?*"), or template-like questions ("*What management successions occurred at IBM in the past year?*").

Class 4. *Interactive QA systems.* These systems are able to answer questions in the context of previous interactions with the user. As reported in Harabagiu et al. [2001], processing a list of questions posed in a context involves complex reference resolution. Unlike typical reference resolution algorithms that associate anaphore with a referent, the reference imposed by context questions requires the association of an anaphora from the current question with either one of the previous questions, answers or their anaphora.

Class 5. *QA systems capable of analogical reasoning.* The characteristic of these systems is their ability to answer speculative questions similar to:

"Is the Fed going to raise interests at their next meeting?"
"Is the US out of recession?"
"Is the airline industry in trouble?"

Since the answer to such questions is probably not explicitly stated in documents, simply because events may not have happened yet, QA systems from this class are decomposed into queries that extract pieces of evidence, after which the answer is formulated using reasoning by analogy. The resources include ad hoc knowledge bases generated from mining text documents clustered by the question topic. Associated with these knowledge sources are case-based reasoning techniques as well as methods for temporal reasoning, spatial reasoning, and evidential reasoning.

While the taxonomy above refers to QA systems, the set of TREC questions can be associated with these system classes since we know now the source of the answer and the system capability required to extract these answers. Thus the system classes and the question classes are used interchangeably in this paper. Table I illustrates the distribution of TREC questions according to the system class required for their processing. In addition to 1393 main-task questions collected from TREC-8, TREC-9, and TREC-2001, there are 25 list questions

Table I. Distribution of TREC Questions

Type	Number (%)
Class 1 (factual)	985 (67.5%)
Class 2 (simple-reasoning)	408 (27.9%)
Class 3 (fusion-list)	25 (1.7%)
Class 4 (interactive-context)	42 (2.9%)
Class 5 (speculative)	0 (0.0%)

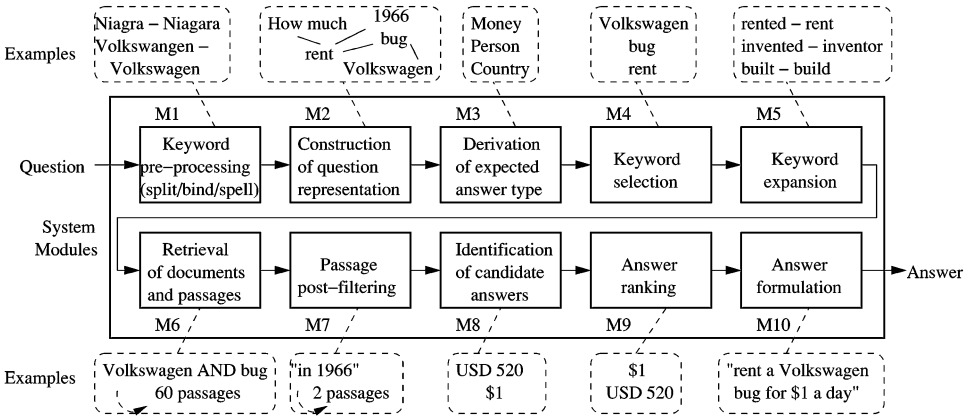

Fig. 1. Architecture of baseline serial system (no feedbacks).

(e.g., *"Name 20 countries that produce coffee."*) and 42 context questions (e.g., *"How long was the Varyag?"*; *"How wide?"*).

3. SERIAL SYSTEM ARCHITECTURE

This section introduces the serialized architecture of our QA system in which there are no feedbacks. The complete architecture with all the feedbacks is presented in a later section of the paper.

3.1 Description of System Modules

As shown in Figure 1, the architecture consists of 10 modules performing several natural language processing tasks.

The first five modules correspond to question processing, the next two modules perform document and passage processing, and the last three modules perform answer processing.

M1. The individual question words are spell-checked. Words like *Volkswangen* and *Niagra* are expanded into their spelling variants *Volkswagen* and *Niagara*. If necessary, questions such as Q885—*"Rotary engine cars were made by what company?"*—are rephrased into a normalized form where the wh-word (*what*) appears at the beginning, for example, *"What company were rotary engine cars made by?"*

M2. The input question is parsed and transformed into an internal representation [Harabagiu et al. 2000] capturing question concepts and binary dependencies between the concepts. Stop words (e.g., prepositions or determiners) are identified and removed from the representation. For illustration, the representation for Q013—*"How much could you rent a Volkswagen bug for in 1966?"*—captures the binary dependency between the concepts *rent* and *1966*.

M3. The mapping of certain question dependencies on a WordNet-based answer type hierarchy disambiguates the semantic category of the expected answers [Paşca and Harabagiu 2001]. For example, the dependency between *How much* and *rent* for Q013 is exploited to derive the expected answer type *Money*. The answer type is passed to subsequent modules for the identification of possible answers (all monetary values).

M4. Based mainly on part of speech information, a subset of the question concepts are selected as keywords for accessing the underlying document collection. A passage retrieval engine accepts Boolean queries built from the selected keywords, for example, *Volkswagen* AND *bug*. The retrieval engine returns passages that contain all keywords specified in the Boolean query. Therefore keyword selection is a sensitive task. If the wrong question word (e.g., *much*) is included in the Boolean query (*much* AND *Volkswagen* AND *bug*), the retrieval is unsuccessful since the passages containing the correct answers are missed.

M5. Before the construction of Boolean queries for actual retrieval, the selected keywords are expanded with morphological, lexical, or semantic alternations. The alternations correspond to other forms in which the question concepts may occur in the answers. For example, *rented* is expanded into *rent*.

M6. The retrieval engine returns the documents containing all keywords specified in the Boolean queries. The documents are then further restricted to smaller text passages where all keywords are located in the proximity of one another. Each retrieved passage includes additional text (extra lines) before the earliest and after the latest keyword match. For illustration, consider Q005—*"What is the name of the managing director of Apricot Computer?"*—and the associated Boolean query *Apricot* AND *Computer* AND *director*. A relevant text fragment from the document collection is *"Dr Peter Horne, managing director of Apricot Computers."* Unless additional text is included in the passages, the actual answer *Peter Horne* would be missed because it occurs before all matched keywords, namely, *director*, *Apricot*, and *Computer*.

M7. The retrieved passages are further refined for enhanced precision. Passages that do not satisfy the semantic constraints specified in the question are discarded [Paşca 2001]. For example, some of the passages retrieved for Q013 do not satisfy the date constraint *1966*. Out of the 60 passages returned by the retrieval engine for Q013, two passages are retained after passage postfiltering.

M8. The search for answers within the retrieved passages is restricted to those candidates corresponding to the expected answer type. If the expected answer type is a named entity, the candidates are identified with a named entity

recognizer. For instance, if the expected answer type is MONEY, the identified candidates include $*1* and *USD 520*. Conversely, if the answer type is a DEFINITION, for example, Q903: *"What is autism?"*—the candidates are obtained by matching a set of answer patterns on the passages. For example the answer *"developmental disorders such as autism"* is extracted with the pattern *"<AP> such as <QP>,"* where <AP> matches the candidate answer and <QP> matches the question phrase *"autism."*

M9. Each candidate answer receives a relevance score according to lexical and proximity features such as distance between keywords, or the occurrence of the candidate answer within an apposition [Paşca 2001]. The candidates are sorted in decreasing order of their scores.

M10. The system selects the candidate answers with the highest relevance scores. The final answers are either fragments of text extracted from the passages around the best candidate answers, or they are internally generated.

3.2 Data Flow Among System Modules

Figure 2 constitutes a detailed view on the information items that are passed among system modules [Paşca 2001]. The submitted question, *"How much could you rent a Volkswagen bug for in 1966?"*, is parsed and then transformed into the question representation, containing question terms connected to each other through binary relations (modules M1-M2). In particular, the relation between the question stem *How much* and the question term *rent* allows for the derivation of the expected answer type, namely, MONEY (module M3). The keyword selection module identifies the available keywords (*Volkswagen*, *bug*, and *rent*) (modules M4-M5).

The available keywords are used for the construction of the Boolean query "Volkswagen AND bug." The keyword *rent* is not included initially in the Boolean query due to its part of speech. The query is passed to the passage retrieval engine. The engine's output is neither too large nor too small—60 text passages are retrieved from the document collection (module M6). If the output were too large, the next available keyword *rent* would be included in the Boolean query "Volkswagen AND bug AND rent," which would be then submitted to the passage retrieval engine.

Because the question term *rent* is deemed to be very specific, all passages which do not contain the term are rejected after the application of the concept specificity filter [Paşca 2001]. One of the rejected passages, from the *Los Angeles Times* article LA070989-0192, mentions a basketball player who *owns* a Volkswagen bug rather than *rents* it (see Figure 2). Similarly, the absence of the date constraint *1966* from any of the text passages causes the elimination of the passage. This is helpful in rejecting passages such as that from article LA010790-0113, which does refer to the rental price of Volkswagen bugs, but not the price from 1966. As a combined effect of the passage postfiltering procedures (module M7), the initial set of 60 passages is reduced to only two passages, both of which are shown in the figure.

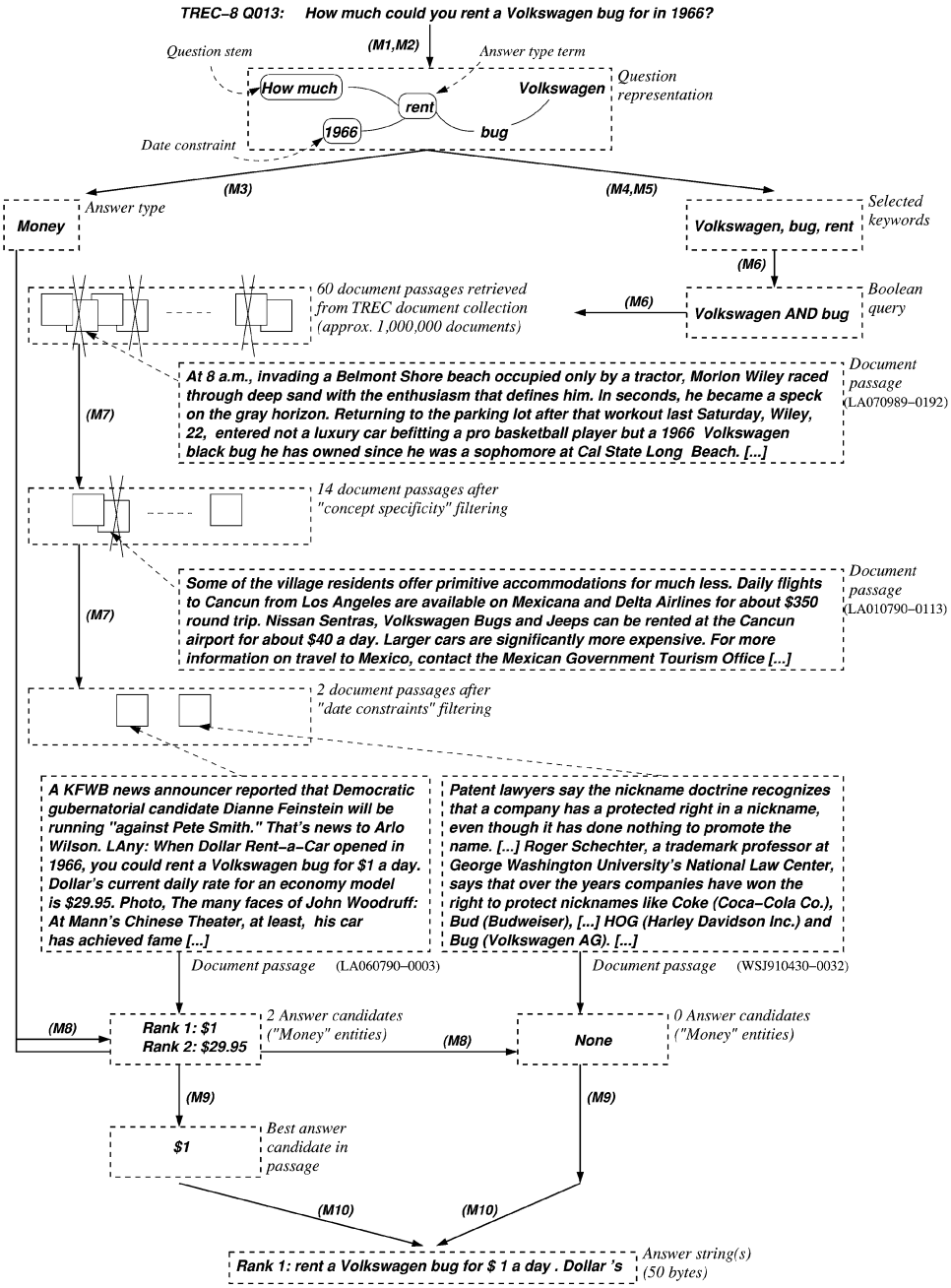

Fig. 2. Data flow for serial system architecture.

The first of the reranked, filtered document passages contains two entities of the expected answer type MONEY, for example, $*1* and $*29.95* (module M8). The first candidate answer according to our ranking metric, $*1*, is also the highest ranking candidate answer (module M9). The returned answer strings are

extracted from the text passages around the highest ranking candidate answer (module M10). Sometimes a passage does not contain any candidate answer; an example is the passage from the *Wall Street Journal* article WSJ910430-0032. In such cases, the position of a "virtual" answer is computed based on the average position of question terms matched in the passage and the answer string extracted accordingly. Since two of the question terms, namely, *Volkswagen* and *bug*, are matched toward the end of the passage from WSJ910430-0032, the corresponding "virtual" answer string is extracted around the average position given by *"and Bug (Volkswagen AG)."*

4. PERFORMANCE PER QUESTION CLASS

4.1 Performance Experiments

The system was tested on 1460 questions collected from TREC-8, 9, and TREC-2001. Answers were extracted from a 3-Gbyte text collection containing about 1 million documents from sources such as the *Los Angeles Times* and *Wall Street Journal*. Each answer has at most 50 characters.

The answer accuracy was measured by the Mean Reciprocal Rank (MRR) metric used by NIST in the TREC QA evaluations [Voorhees 1999]. The reciprocal ranking basically assigns a number equal to $1/R$ where R is the rank of the correct answer. Only the first five answers are considered; thus R is less or equal to 5. When the system does not return a correct answer in the top five, the precision score for that question is zero. The overall system precision is the mean of the individual scores. System answers were measured against correct answer keys, with an evaluation script implementing the MRR metric. Both the answer keys and the script were provided by NIST.

4.2 Performance per Question Stem

Question stems (*Where*, *What*, *Who*, etc.) provide one of the simplest and yet useful classifications of factual questions. The role of the question stems is twofold. First, they differentiate between declarative (*"In 1966, you could rent a Volkswagen bug for $1 a day"*) and interrogative statements (*"How much could you rent a Volkswagen bug for in 1966?"*). Second, they are used in many QA system implementations as (coarse) clues in the identification of the expected answer type (cf. Gaizauskas and Humphreys [2000]; Srihari and Li [2000]). Thus *Who* questions often (but not always!) ask about *Person* names, *Where* questions refer to locations, and so forth.

Table II illustrates the relation between the precision of our QA system and all question stems appearing in the 1393 TREC main-task questions. The large majority of TREC questions have the *What*, *Who*, *Where*, and *When* question stems, with *What* questions constituting more than half of the evaluation questions. In turn, *What* questions can be often subcategorized into one of the other question stems. For instance *"What was the name of the Titanic's captain?"* corresponds to *Who*; similarly *"What country is the biggest producer of tungsten?"* corresponds to *Where*. The most common *What* subcategories are *Where* (17%), modified *How* (*How far*, *How many*, *How much* etc.) (8%), and *Who* (5%).

Table II. Answer Accuracy Across Various Question Stems

Question stem	Number of questions	Precision (MRR)	Question stem	Number of questions	Precision (MRR)
Where	131	0.657	How much	15	0.350
How far	6	0.639	How tall	6	0.333
Whom	3	0.583	How fast	4	0.312
Who	204	0.550	How big	2	0.250
Name	35	0.545	How hot	2	0.250
How wide	2	0.500	How large	2	0.250
When	92	0.493	How long	8	0.198
What	764	0.430	How old	5	0.040
How many	57	0.429	How rich	1	0.000
How	7	0.429	Why	6	0.000
Which	21	0.381	(Unknown)	2	0.000

The answer accuracy has a wide variation across the question stems. At the higher end, *Where* questions are answered with a precision score of 0.657. Comparatively, none of the six *Why* questions have correct answers, which lends support to the notion that questions requiring advanced reasoning mechanisms are more difficult. With the exception of two questions, the system finds exactly one question stem for each question. The system fails to properly identify the question stem for Q577—*"Can you give me the name of a clock maker in London, England?"*—and Q984—*"The U.S. Department of Treasury first issued paper currency for the U.S. during which war?"*

4.3 Performance per Answer Type and Question Class

The classification of questions according to their stems has a few limitations. Not only can ambiguous question stems (*What*, *Which*) be associated with almost any possible answer type (*What king* . . . , *What country* . . . , *What city* . . . , *What color* . . . , etc.), but it is also possible to formulate equivalent questions by using different question stems. Consider the questions Q803—*"What king signed the Magna Carta?"*—and Q804—*"Who was the king who signed the Magna Carta?"* While the question stems are different (*What* vs. *Who*), the expected answer types are identical (a *Person* name).

An alternative analysis of the system performance on the main-task questions, across answer types rather than question stems, is shown in Table III. The values reflect the answer types actually recognized by the system rather than the ideal answer types that would be assigned by a human. The answer type is unknown when the system fails to identify it. Ideally, each question would have exactly one answer type if the question and the user's intentions were fully understood. In practice, the QA system identifies zero (i.e., unknown answer type), one (i.e., nonambiguous answer type), or more answer types for each question. On average there are 1.5 expected answer types identified per TREC question. Questions with several expected answer types are accounted for in more than one entry in Table III to conserve their ambiguity. Note that other approaches also allow for multiple answer types per question [Abney et al. 2000]. For instance, Q1359—*"What do you call a word that is spelled the same backwards and forwards?"*—has zero expected answer types (the answer

Table III. Answer Accuracy Across Various Answer Types

Answer type	Number of questions	Precision (MRR)	Answer type	Number of questions	Precision (MRR)
Airport Code	1	1.000	Person	225	0.499
Attraction	7	1.000	Date	143	0.465
Phone Number	1	1.000	Mammal	13	0.423
Author	13	0.769	Number	80	0.409
Currency	7	0.750	Manner (How)	8	0.406
Organization	33	0.733	Product	34	0.386
Plant	8	0.719	Quantity	86	0.383
City	166	0.676	Nationality	12	0.354
Continent	120	0.665	Percent	17	0.301
University	115	0.659	Price	27	0.252
Chemical Elem.	8	0.656	Money	28	0.243
Province	135	0.640	Instrument	7	0.179
Country	147	0.633	Color	12	0.132
Disease	11	0.632	(Unknown)	228	0.132
Other Location	151	0.618	Time	2	0.125
Definition	171	0.613	Reason (Why)	8	0.031
Language	7	0.607	Address	1	0.000
Quote	9	0.528			

Table IV. Answer Accuracy per Question Class

Type	Precision (MRR)
Class 1 (factual)	0.641
Class 2 (simple-reasoning)	0.406
Class 3 (fusion-list)	0.760
Class 4 (interactive-context)	0.729
Class 5 (speculative)	N/A

type is unknown), and Q035—*"What is the name of the highest mountain in Africa?"*—has one answer type, *Other Location* (a location name that is not a country, province, city, continent, or university). The internal answer type for Q163—*"What state does Charles Robb represent?"*—is multiple; both *Province* (which includes U.S. states such as *Virginia* or *Texas*) and *Country* (such as *USA* or *Canada*) are retained.

Table III groups into the same entry those answer types that are mapped into a common named entity category. For example the entry for *Quantity* groups subtypes such as *Distance* (*3.2 miles*), *Duration* (*2 hours*), and *Size* (*2.3 square feet*). Comparatively, the entry for *Number* corresponds only to simple numbers not followed by measurement units.

The 1393 main-task questions discussed above are part of classes 1 and 2 in Table IV. Most questions were selected from actual search engine logs. Comparatively, the evaluation questions of classes 3 (list) and 4 (context) were created by NIST assessors. In particular, Voorhees [2001] indicated that context questions were easier to answer than originally intended. Moreover, the question context usually improves the accuracy of passage retrieval as more keywords become available. Class 5 is not represented in the TREC evaluation questions.

Table V. Distribution of Errors per System Module

Module	Module definition	Errors (%)
(M1)	Keyword preprocessing (split/bind/spell check)	1.9
(M2)	Construction of internal question representation	5.2
(M3)	Derivation of expected answer type	36.4
(M4)	Keyword selection (incorrectly added or excluded)	8.9
(M5)	Keyword expansion desirable but missing	25.7
(M6)	Actual retrieval (limit on passage number or size)	1.6
(M7)	Passage postfiltering (incorrectly discarded)	1.6
(M8)	Identification of candidate answers	8.0
(M9)	Answer ranking	6.3
(M10)	Answer formulation	4.4

5. ERROR ANALYSIS FOR THE BASELINE SERIAL SYSTEM

5.1 Aggregated Module Errors

The inspection of internal traces system errors for each evaluation question. The goal in this experiment is to identify the earliest module in the chain (from left to right) that prevents the system from finding the right answer, that is, causes the error.

As shown in Table V, question preprocessing is responsible for 7.1% of the errors distributed among module M1 (1.9%) and M2 (5.3%). Most errors in module M2 are due to incorrect parsing (4.5%). Two of the ten modules (M3 and M5) account for more than half of the errors. The failure of either module makes it hard (or impossible) for subsequent modules to perform their task. Whenever the derivation of the expected answer type (module M3) fails, the set of candidate answers identified in the retrieved passages is either empty in 28.2% of the cases (when the answer type is unknown) or contains the wrong entities for 8.2% (when the answer type is incorrect). If the keywords used for passage retrieval are not expanded with the semantically related forms occurring in the answers (module M5), the relevant passages are missed.

The selection of keywords from the internal question representation (module M4) coupled with the keyword expansion (module M5) generates 34.6% of the errors. Both these modules affect the output of passage retrieval, since the set of retrieved passages depends on the Boolean queries built and submitted to the retrieval engine by the QA system.

Modules M6 and M7 are responsible for the retrieval of passages where answers may actually occur. Their combined errors is 3.2%. In module M6, there are parameters to control the number of retrieved documents and passages, as well as the size of each passage.

Answer processing is done in modules M8 through M10. When the expected answer type is correctly detected, the identification of the candidate answers (module M8) produces 8.0% errors. 3.1% errors are due to named entity recognition (the "incomplete dictionaries" should be erased) and 4.9% are due to spurious answer pattern matching. Modules M9 and M10 fail to rank the correct answer within the top five returned in 10.7% of the cases. Module M9 fails if the correct answer candidate is not ranked within the top five, whereas M10

fails if the returned answer string is incomplete, namely, it does not fit within 50 bytes.

5.2 Examples of Module Errors

In addition to the aggregated errors presented above, this section provides a more detailed view on particular errors that occur within the QA system. One question is selected as an example for each system module.

M1. The spell checker fails to recognize the variation *Niagra* in the question Q589: *"What state is Niagra Falls located in?"* The relevant text fragments of the document collection, *"across the Niagara Falls, in New York state"* or *"Niagara Falls, N.Y., also has a built-in price advantage,"* all use the spelling *Niagara*. None of the relevant text fragments are identified because the Boolean queries use the variation *Niagra* rather than *Niagara*.

M2. Errors in the part of speech tagger affect the construction of the question representation. For Q250—*"Where did the Maya people live?"*—the term *live* is tagged NN (noun) instead of verb. The incorrect tag makes *live* a higher-priority keyword. The term is included in the Boolean query used to search the document collection. Therefore relevant text fragments such as *"There were plenty of rooms at the Mayan site [. . .] at Santa Rosa Xtampak , an ancient regional capital in the heart of the Yucatan"* are missed during retrieval because they do not contain the keyword *live*. The system returns incorrect 50-byte answers, the first of which is *"people live within Liverpool Airport's."*

M3. The extraction of incorrect answers for Q518—*"In what area of the world was the Six Day War fought?"*—is due to an error in module M3. The system determines the answer type of the question from the question term *area*. The question is asking about a *Location* name (some area of the world). The system incorrectly classifies *area* under the *Quantity* category. Note that the classification would be correct for a question such as *"What is the area of a football field?"* The extracted answer strings are centered around quantities rather than location names.

M4. The inclusion of the wrong keyword in the Boolean queries leads to the retrieval of irrelevant passages. The selection of the keyword *place* from the phrase *take place* is incorrect since the collocation *take place* means *happen* or *occur*, and that meaning would be altered if terms were considered separately. The term *place* from the question Q186—*"Where did the Battle of the Bulge take place?"*—is incorrectly included in the Boolean query, Battle AND Bulge AND *place*. Relevant text fragments are missed because they do not make any direct reference to *place*; examples of relevant fragments are *"Luxembourg has two other Battle of the Bulge museums"* and *"Germans captured the two Dec. 19, 1944, during the Battle of the Bulge in southeastern Belgium, northern Luxembourg and northern France."*

M5. In the case of Q1236—*"What is the murder rate in Windsor, Ontario?"*—the system fails to expand the keyword *murder* with its synonyms, *homicide*

and *slaying*. Therefore none of the retrieved passages captures a relevant text fragment, that is, *"for Windsor [. . .] the three homicides in Canada's City of Roses last year translate to about 1.5 per 100,000 people."*

M6. The first 50-byte answer returned for Q1275—*"Who was the first vice president of the U.S.?"*—is incorrect: *"and U.S. Vice President Al Gore. Sir Leon."* Even though the system includes all available keywords in the Boolean query U.S. AND first AND vice AND president, the output from passage retrieval is too large. Out of the 600 retrieved passages, only the top 500 are actually searched for an answer. The relevant answer, *John Adams*, appears in a passage at the end of the retrieved list, and therefore it is missed.

M7. After passage retrieval, the system incorrectly discards all passages that do not contain the term *year* for Q387: *"What year did Montana become a state?"* The relevant text fragments are thus missed and the first returned answer, *1990*, is incorrect.

M8. An error in module M8 occurs for Q120: *"Who held the endurance record for women pilots in 1929?"* Modules M1-M7 perform correctly. The identified answer type is *Person* (module M3). After keyword expansion, the Boolean query is: endurance AND record AND (women OR woman) AND (pilots OR pilot) (modules M4, M5). One of the eight passages retrieved (modules M6, M7) contains the correct answer *Trout*. The relevant text fragment is *"For Trout , making her own rules and regulations has won her a place in history. She held the women's pilot endurance record of 12 hours, 11 minutes in 1929 [. . .] Trout's name is engraved in the Women's Hall of Fame [. . .] said archivist Ray Wagner."* When the process reaches module M8, it fails because the named entity recognizer does not capture *Trout* as a person name. Consequently another name (*Ray Wagner*) is selected from the same passage and the extracted answer string is incorrect.

M9. The correct answer for Q536—*"What is the population of the United States?"*—is returned only at rank 6, due to the higher scores internally assigned to the other five candidates. The surrounding context for the sixth, correct answer is *"A spot in heavily wooded land in Missouri's Crawford County is the population center of the United States [. . .] an imaginary, flat, weightless and rigid map of the nation would balance perfectly if all 248,709,873 residents [. . .]"*. The question keywords, *population*, *United* and *States* are relatively far from the numeral *248,709,873* so the answer is assigned a lower score. Comparatively the first, incorrect answer, *"1986) United States 15.5 France 14.1 United,"* is part of the fragment *"A Statistical Portrait Birthrates (Live births per 1,000 population, 1986) United States 15.5 France 14.1 United Kingdom 13.3."* The question keywords are grouped closer together for the first answer and therefore they are assigned a higher score.

M10. The answer strings are sometimes slightly shifted from the ideal strings and thus become irrelevant due to near-misses. An example is the answer string *"not. Metabolism is the process of breaking down"* for the question

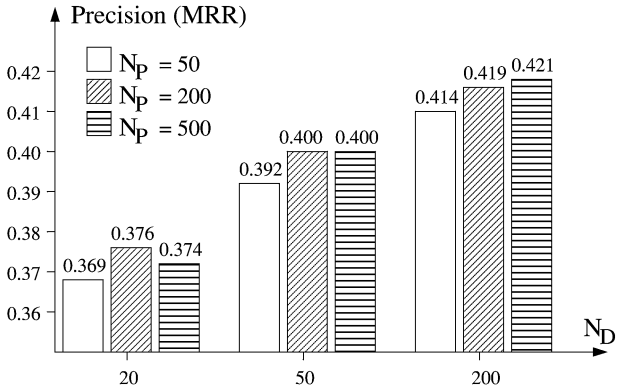

Fig. 3. Impact of maximum number of documents and passages processed.

Q1303: *"What is metabolism?"* If shifted to the right, the answer would become relevant: *"Metabolism is the process of breaking down food."*

6. IMPACT OF RETRIEVAL PARAMETERS

The quantitative performance of the QA system is largely dependent on the amount of text retrieved from the document collection; the more text is retrieved, the better the chance of finding the answer. However, practical QA systems cannot afford to apply time-consuming NLP techniques (especially parsing) to very large amounts of text. Retrieval parameters are used to provide trade-offs between the amount of text passed into the answer processing module and the accuracy of the extracted answers. The QA system has the following *retrieval parameters*:

—N_D: the maximum number of documents retrieved from a subcollection (default value 200 for each of 14 sub-collections)[1];

—N_P: the maximum number of passages processed to identify candidate answers (default value 500);

—S_P: the size allowance for each retrieved passage (default value 10 lines before the earliest and after the latest keyword match).

When N_D and N_P are set to smaller values, the execution time is lower but relevant documents and passages may be missed. Figure 3 illustrates the impact of the parameters N_D and N_P on the precision computed over the entire set of 1460 test questions. The higher the number of documents retrieved, the higher the precision score. It is apparent that N_P has a relatively smaller impact on the precision than N_D. This is due to the fact that the retrieved passages are reordered based on a set of lexical features, such that the identification of the candidate answers is performed on the top N_P reordered passages.

Figure 4 shows the possible trade-off between overall precision and execution time as a function of the passage size S_P. It also provides details on the

[1]The passage retrieval engine manages the TREC document collection as a set of 14 separate subcollections.

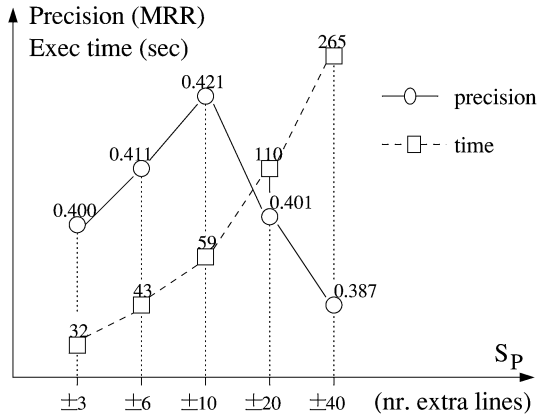

Fig. 4. Impact of passage size on precision and execution time.

efficiency of our system. Interestingly, the highest precision score occurs for the default setting, ±10. When S_P is smaller, the answers are missed because they do not fit in the retrieved passages. When S_P is larger, the actually relevant text fragments are submerged in a large amount of text. Consequently the answer ranking module (M9 from Figure 1) sorts through a very large number of candidate answers, and it does not always rank the correct answers within the top five returned.

7. BOOLEAN VERSUS VECTOR-SPACE DOCUMENT RETRIEVAL

In addition to the methods used for passage retrieval (cf. Salton et al. [1993]) and the values of the retrieval parameters, the quality of the fragments retrieved from the collection also depends on the document retrieval model. For the purpose of document retrieval, most QA systems use the Boolean model [Harabagiu et al. 2001] or the statistical or vector-space model [Abney et al. 2000; Hovy et al. 2001]. This section briefly describes the two models and compares their impact on the overall answer accuracy of our QA system.

In both models, the document retrieval engine compares the query against the document collection and retrieves those documents that match the query. The criteria used to decide if a query matches a document (in other words, if a document is relevant for a given query) differ from a model to another. In the simple Boolean model [Salton and McGill 1983], queries consist of terms linked by Boolean operators, for example, (Japanese OR German) AND (car OR automobile). A document is relevant for the query if the document satisfies the query logical expression. The relevance score assigned to the document is either 0 (irrelevant) or 1 (relevant). The simple Boolean model does not offer any further ranking among relevant documents. Various hybrid Boolean models were developed to alleviate the lack of document ranking, but they are difficult to apply in practice [Savoy 1997].

In the case of the vector-space model, documents and queries are represented as vectors whose dimensionality is the size of the vocabulary, that is, the number of distinct words from the entire collection [Salton and Buckley 1988]. The

Table VI. Impact of Boolean Versus Vector-Space Document Retrieval on QA Precision for TREC-8 Questions

N_D = Max. number of documents retrieved from the entire collection
P's = Average nr. of retrieved passages
Q's = Nr. questions with some correct answer in top 5

Retrieval model	N_D	P's	Precision (MRR)	Q's
Boolean	2800	262	0.578	141/200
Vector (cosine)	50	74	0.373	100/200
Vector (in-house)	50	42	0.483	121/200
Vector (in-house)	200	96	0.553	135/200
Vector (in-house)	300	120	0.577	145/200
Vector (in-house)	500	151	0.582	145/200

vectors store the weights of the terms, for example, w_{d_k} is the weight of the kth term of the document vector D. The weights are generally statistical measurements of the importance of a term within the document. In these experiments, the weights of the query terms are 1 and the weights of the document terms are the term frequency within the document, for example, 4 if the term occurs four times in the document. When the query is matched on a document, a similarity score is computed by comparing the query vector and the document vector. The retrieved documents are ranked in decreasing order of their similarity score. The best-known similarity formula in vector-space document retrieval is the cosine formula [Salton and Buckley 1988], which computes the inner product of the two vectors:

$$\text{sim}(Q, D) = \frac{\sum_{k=1}^{n}(w_{q_k} \times w_{d_k})}{\sqrt{\sum_{k=1}^{n}(w_{q_k})^2 \star \sum_{k=1}^{n}(w_{d_k})^2}},$$

where n is the vocabulary size, Q is the query vector, D is the document vector, w_{q_k} is the weight of the kth term of Q and w_{d_k} is the weight of the kth term of D. In the experiments reported here, the value of the denominator was ignored.

The first part of Table VI compares the answer accuracy respectively with Boolean retrieval and vector-space retrieval using the cosine similarity. Note that the experiments in Table VI are performed on TREC-8 questions (Q001-Q200) with $N_P = 500$ (number of passages processed) and $S_P = \pm 10$ (passage size allowance). Also note that the Boolean engine returns at most the specified maximum number of documents N_D, or usually less as fewer documents actually match the query. In contrast, the vector-space engine always returns the specified maximum number of documents, by retrieving the N_D documents with the highest similarity scores.

The cosine formula (or one of its variations) is frequently used to identify full-length documents that match a specified query [Salton and Buckley 1988; Salton and McGill 1983]. However, our experiments show that it is not necessarily the best similarity formula in the context of QA, where the goal is to identify brief answers. Consider one of the TREC-8 questions, namely, Q008: *"What is the name of the rare neurological disease with symptoms such as: involuntary movements (tics), swearing, and incoherent vocalizations (grunts, shouts, etc .)?"*

The simplified query for this question is (rare AND neurological AND disease AND involuntary AND movements AND incoherent AND vocalizations). The query vector is compared with all TREC documents, in particular with three documents from the *Los Angeles Times*. The numerator of the similarity formula becomes

$$\text{numer}(\text{sim}(Q_8,\text{LA091789-0180})) = 1 \times 22(\textit{movement}) = 22,$$

$$\begin{aligned}\text{numer}(\text{sim}(Q_8,\text{LA092689-0119})) = {} & 1 \times 38(\textit{disease}) + 1 \times 2(\textit{rare}) \\ & +1 \times 1(\textit{neurological}) = 41,\end{aligned}$$

$$\begin{aligned}\text{numer}(\text{sim}(Q_8,\text{LA121690-0224})) = {} & 1 \times 1(\textit{movement}) + 1 \times 1(\textit{disease}) \\ & + 1 \times 1(\textit{involuntary}) + 1 \times 1(\textit{rare}) \\ & + 1 \times 1(\textit{neurological}) + 1 \times 1(\textit{incoherent}) \\ & = 6.\end{aligned}$$

The first two documents have a high similarity score because one of the query terms (*movement* and *disease*, respectively) is matched many times (22 and 38 times, respectively) in the documents. Comparatively, the third document has a lower score; in fact, more than 200 other TREC documents have a higher score than LA121690-0224. The similarity scores are in total contradiction with the actual relevance of the documents. None of the documents LA091789-0180 and LA092689-0119 is relevant for Q008, whereas LA121690-0224 contains the relevant fragment: *"she has both Tourette's Syndrome and obsessive-compulsive disorder. The syndrome is a rare neurological disease with a variety of symptoms, including involuntary purposeless movements, swearing, tics and incoherent grunts and barks."*

The lesson learned is that, in the context of QA, the occurrence of many query terms in the document is a better relevance indicator than the frequent occurrence of only one query term. Hence we developed formulas that prioritize the number of keywords versus their frequency. A similarity formula that takes this factor into account is

$$\text{sim}(Q, D) = \sum_{k=1,(w_{q_k} \times w_{d_k} != 0)}^{n} (T + w_{q_k} \times w_{d_k}). \tag{1}$$

The factor T is collection-dependent (it is set to 50 in the experiments). As shown in Table VI, the new, in-house similarity formula performs better than the cosine similarity. The overall answer accuracy increases as the number of documents retrieved increases. For 500 documents retrieved per question, both the precision score and the number of correctly answered questions are slightly higher than in the case of Boolean retrieval. Note that the answer accuracy values shown in Table VI for the vector-space model could be higher if we used one of the more recent methods for weight computation.

8. IMPACT OF FEEDBACKS

The results presented in previous sections correspond to the serialized baseline architecture from Figure 1. That architecture is in fact a simplified version of our system which uses several feedbacks to boost the overall performance.

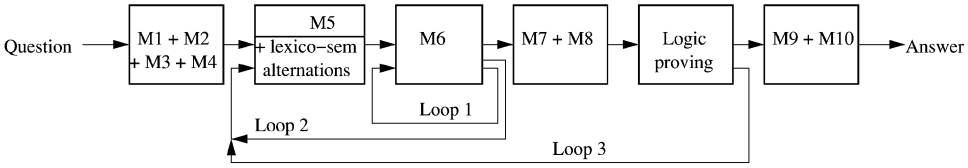

Fig. 5. Architecture with feedbacks.

As shown in Figure 5, the architecture with feedbacks extends the serialized architecture in several ways. Keyword expansion (module M5) is enhanced to include lexico-semantic alternations from WordNet. A new module for logic proving and justification of the answers is inserted before answer ranking.

Logic proving relies on the transformation of the representations of the question and answer into logic forms (first-order logic formulas) [Harabagiu et al. 2000]. For example, the question Q006—*"Why did David Koresh ask the FBI for a word processor?"*—can be represented with the following logic form:

$$\text{REASON}(x_1) \wedge \text{David}(x_2) \wedge \text{Koresh}(x_2) \wedge \text{ask}(e, x_1, x_2, x_3, x_4) \wedge \text{FBI}(x_3) \wedge \text{word}(x_4) \wedge \text{processor}(x_4) \wedge \text{HUMAN}(x_2) \wedge \text{ORGANIZATION}(x_3).$$

In the logic forms, the predicate arguments encode interterm dependencies. Predicates that belong to the same concept (e.g., *David*, *Koresh*) take the same arguments (x_2). The verbal predicate *ask* links the nominal arguments around the event e of *ask*ing. Note that the logic form also encodes semantic information: *David Koresh* is a Person, *FBI* is an Organization, and the expected answer type is *Reason*.

The logic prover operates on the question logic form and answer logic form, trying to find a proof of the question from the answer. If a proof is found, the answer is correct; otherwise, the answer is incorrect and should be rejected. More details regarding the logic representation of text and the prover are presented in Moldovan et al. [2002].

In addition to the new modules, three loops become an integral part of the system: the passage retrieval loop (loop 1); the lexico-semantic loop (loop 2); and the logic proving loop (loop 3).

As part of loop 1 (Figure 5), the Q/A system adjusts Boolean queries before passing them to the retrieval engine. The system verifies that the output is not too large or too small by checking whether the number of retrieved passages is between two collection-dependent thresholds. If the output from the retrieval engine is too small, a keyword is dropped and retrieval resumed. If the output is too large, a keyword is added and a new iteration started, until the output size is neither too large nor too small. When lexico-semantic connections from the question to the retrieved passages are not possible, loop 2 is triggered. Question keywords are replaced with WordNet-based alternations and retrieval is resumed. Loop 3 relies on the logic prover that verifies the unifications between the question and logic forms. When the unifications fail, the keywords are expanded with semantically related alternations and retrieval is repeated.

Table VII. Impact of Feedbacks on Precision

Feedback added	Precision (MRR)	Incremental enhancement
None	$0.421 = b$	0%
Passage retrieval (loop 1)	$0.468 = b_1$	$b + 11\%$
Lexico-semantic (loop 2)	$0.542 = b_2$	$b_1 + 15\%$
Proving (loop 3)	$0.572 = b_3$	$b_2 + 5\%$

Table VII illustrates the impact of the retrieval loops on the answer accuracy. The knowledge brought into the question answering process by lexico-semantic alternations has the highest individual contribution, followed by the mechanism of adding/dropping keywords.

9. IMPACT OF NATURAL LANGUAGE PROCESSING ON QA

The availability of an array of natural-language modules and resources (parsers, WordNet, answer type hierarchies, named entity recognizers, question and answer semantic transformations, and so forth) enables trade-offs between answer processing complexity and answer accuracy. Table VIII shows the overall precision for four different settings of answer processing.

9.1 Impact of Answer Processing Complexity

The first setting, *direct extraction*, corresponds to the simplest QA system that does not use any NLP techniques or resources. No attempt is made to estimate the location of the relevant text fragments within the retrieved passages. Instead, all answer strings are extracted from the start of each passage, and returned in the order in which the passages were retrieved. The answer precision is only 0.028.

When the NLP techniques are enabled, with the exception of the derivation of the expected answer type (module M8), the answer accuracy is still limited. The candidate answers cannot be properly identified without knowing their semantic category (persons, cities, and so forth). With no semantic information available, the answer ranking module (module M9) estimates the position of relevant text fragments within the passages. The estimation relies on the proximity-based lexical features computed among question terms matched in the passages. Higher values of the features imply higher relevant for the corresponding text fragment. The passages are reranked before answer formulation. The precision improves from 0.028 for direct extraction to 0.150 for lexical matching.

If the derivation of the expected answer type is also enabled, the precision score changes to 0.468. The system extracts the answers around entities that match the expected answer type, for example, person names if the answer type is *Person*. Finally, when all feedbacks are enabled, the highest overall precision of 0.572 is achieved. Comparatively, the answer processing modules of other QA systems usually span over levels 2 and 3 from Table VIII.

The final precision scores for TREC-8, TREC-9, and TREC-2001 are, respectively, 0.555, 0.580, and 0.570. Therefore the precision did not vary much in

Table VIII. Performance of Answer Processing

Answer processing complexity level	Modules used	Precision (MRR)
(1) Direct extraction	M1-M6, M10	0.028
(2) Lexical matching	M1-M7, M9-M10	0.150
(3) Semantic matching	M1-M10	0.468
(4) Feedbacks enabled	All	0.572

spite of the higher degree of difficulty. This is due to the increased use of natural language processing (NLP) in our system.

9.2 Impact of Resource Usage

The second set of experiments consists of disabling the main natural language resources used in the QA system, namely, the access to WordNet and the named entity recognizer, to assess their impact on the overall answer accuracy. Note that the parser is an integral part of our question processing model and therefore it is impractical to disable it.

Denote with *b* the baseline system performance when all resources are enabled. The precision score (MRR) drops to *0.59b* if WordNet is disabled. The derivation of the answer type (module M3) and keyword expansion (module M5) from Figure 1 are the two modules that are most influenced by WordNet. For example, the WordNet noun hierarchies specify that the concept *pilot* is a specialization of *aviator*, which in turn is a kind of *person*. The answer type for Q037—*"What was the name of the US helicopter pilot shot down over North Korea?"*—is *Person*. The system cannot derive the answer type correctly unless it has access to WordNet hierarchies because the ambiguous question stem *What* alone does not provide any clue as to what the expected answer type is. A closer analysis shows that the performance drop is more significant for the *What* questions. When WordNet is disabled, the MRR for the *What* questions drops to *0.37b* as compared to *0.59b* for the entire set. This result indicates that the availability of lexico-semantic information becomes more important for difficult questions.

By disabling the named entity recognizer, the answer processing lacks the semantic information necessary to identify candidate answers. Loose approximations for the candidate answers are computed based strictly on keywords matching. In this case the precision drops to *0.32b*.

10. CONCLUSIONS

The main conclusion is that the overall performance of QA systems is directly related to the depth of NLP resources. It also depends on the tools used for answer finding. As shown in Table VIII, the performance of information retrieval techniques is significantly enhanced when lexico-semantic information is fully exploited throughout the answer finding process.

Table V illustrates that the performance bottlenecks of our QA system are due to two modules, namely, the derivation of the expected answer type and the keyword expansion. The bottlenecks are not specific to our QA system but reflect the limitations of current QA technologies. Question answering systems

perform better when the relevant passages and the candidate answers are clearly defined in the questions. The main problem is the lack of powerful schemes and algorithms for modeling complex questions in order to derive as much information as possible, and for performing a well-guided search through thousands of text documents.

The lexico-semantic information imported in the QA system through the retrieval feedbacks brings consistent improvements over serial processing. Per-component errors are spread uniformly over the first four classes of question complexities, indicating how our system improved over the years.

REFERENCES

ABNEY, S., COLLINS, M., AND SINGHAL, A. 2000. Answer extraction. In *Proceedings of the 6th Applied Natural Language Processing Conference* (ANLP-2000, Seattle, WA). 296–301.

BRECK, E., BURGER, J., FERRO, L., HIRSCHMAN, L., HOUSE, D., LIGHT, M., AND MANI, I. 2000. How to evaluate your question answering system every day . . . and still get real work done. In *Proceedings of the 2nd Conference on Language Resources and Evaluation* (LREC-2000, Athens, Greece). 1495–1500.

BRECK, E., LIGHT, M., MANN, G., RILOFF, E., BROWN, B., ANAND, P., ROOTH, M., AND THELEN, M. 2001. Looking under the hood: Tools for diagnosing your question answering engine. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics Workshop on Open-Domain Question Answering* (ACL-01, Toulouse, France). 1–8.

CARDIE, C., NG, V., PIERCE, D., AND BUCKLEY, C. 2000. Examining the role of statistical and linguistic knowledge sources in a general-knowledge question-answering system. In *Proceedings of the 6th Applied Natural Language Processing Conference* (ANLP-2000, Seattle, WA). 180–187.

CLARKE, C., CORMACK, G., LASZLO, M., LYNAM, T., AND TERRA, E. 2002. The impact of corpus size on question answering performance. In *Proceedings of the 25th ACM Conference on Research and Development in Information Retrieval, Poster session* (SIGIR-2002, Tampere, Finland). ACM Press, New York, NY, 367–368.

CLARKE, C., CORMACK, G., AND LYNAM, T. 2001. Exploiting redundancy in question answering. In *Proceedings of the 24th ACM Conference on Research and Development in Information Retrieval* (SIGIR-2001, New Orleans, LA). ACM Press, New York, NY, 358–365.

GAIZAUSKAS, R. AND HUMPHREYS, K. 2000. A combined IR/NLP approach to question answering against large text collections. In *Proceedings of the 6th Content-Based Multimedia Information Access Conference* (RIAO-2000, Paris, France). 1288–1304.

HARABAGIU, S., MOLDOVAN, D., PAŞCA, M., SURDEANU, M., MIHALCEA, R., GÎRJU, R., RUS, V., MORĂRESCU, F. L. P., AND BUNESCU, R. 2001. Answering complex, list and context questions with lcc's question-answering server. In *Proceedings of the 10th Text REtrieval Conference* (TREC-2001). NIST, Gaithersburg, MD. 355–361.

HARABAGIU, S., PAŞCA, M., AND MAIORANO, S. 2000. Experiments with open-domain textual question answering. In *Proceedings of the 18th International Conference on Computational Linguistics* (COLING-2000, Saarbrucken, Germany). 292–298.

HOVY, E., GERBER, L., HERMJAKOB, U., LIN, C., AND RAVICHANDRAN, D. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the Human Language Technology Conference* (HLT-2001, San Diego, CA). 339–345.

ITTYCHERIAH, A., FRANZ, M., ZHU, W., AND RATNAPARKHI, A. 2001. Question answering using maximum-entropy components. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics* (NAACL-2001, Pittsburgh, PA). 33–39.

LIGHT, M., MANN, G., RILOFF, E., AND BRECK, E. 2001. Analyses for elucidating current question answering technology. *Nat. Lang. Eng.* (Special Issue on Question Answering). *7*, 4, 325–342.

MOLDOVAN, D., HARABAGIU, S., GÎRJU, R., MORĂRESCU, P., LĂCĂTUŞU, F., NOVISCHI, A., BĂDULESCU, A., AND BOLOHAN, O. 2002. Lcc tools for question answering. In *Proceedings of the 11th Text REtrieval Conference* (TREC-2002). NIST, Gaithersburg, MD, 144–155.

PAŞCA, M. 2001. High-performance, open-domain question answering from large text collections. Ph.D. thesis, Southern Methodist University, Dallas, TX.

PAŞCA, M. AND HARABAGIU, S. 2001. The informative role of WordNet in open-domain question answering. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics, Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*. (NAACL-01, Pittsburgh, PA). 138–143.

PRAGER, J., BROWN, E., CODEN, A., AND RADEV, D. 2000. Question answering by predictive annotation. In *Proceedings of the 23rd International Conference on Research and Development in Information Retrieval* (SIGIR-2000, Athens, Greece). 184–191.

SALTON, G., ALLAN, J., AND BUCKLEY, C. 1993. Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th ACM Conference on Research and Development in Information Retrieval* (SIGIR-93, Pittsburgh, PA). 49–58.

SALTON, G. AND BUCKLEY, C. 1988. Term-weighting approaches in automatic text retrieval. *Informa. Proc. Manage. 24*, 5, 513–523.

SALTON, G. AND MCGILL, M. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY.

SAVOY, J. 1997. Ranking schemes in hybrid boolean systems: A new approach. *J. Amer. Soc. Inform. Sci. 48*, 3 (June), 235–253.

SRIHARI, R. AND LI, W. 2000. A question answering system supported by information extraction. In *Proceedings of the 6th Applied Natural Language Processing Conference* (ANLP-2000, Seattle, WA). 166–172.

VOORHEES, E. 1999. The TREC-8 Question Answering track report. In *Proceedings of the 8th Text REtrieval Conference* (TREC-8). NIST, Gaithersburg, MD, 77–82.

VOORHEES, E. 2001. Overview of the TREC 2001 Question Answering track. In *Proceedings of the 10th Text REtrieval Conference* (TREC-2001). NIST, Gaithersburg, MD, 42–51.

VOORHEES, E. AND TICE, D. 2000. Building a question-answering test collection. In *Proceedings of the 23rd International Conference on Research and Development in Information Retrieval* (SIGIR-2000, Athens, Greece). 200–207.

Received October 2002; revised December 2002; accepted January 2003