Question Answering with QACTIS at TREC-2004

P. Schone, G. Ciany, P. McNamee†, J. Mayfield†, T. Bassi, A. Kulman*

U.S. Department of Defense
Ft. George G. Meade, MD 20755-6000

ABSTRACT

We provide a description of the QACTIS question-answering system and its application to and performance in the 2004 TREC question-answering evaluation. Since this was also QACTIS's first year competing at TREC, we provide a complete overview of its purpose, development, structure, and its future directions.

1. INTRODUCTION

Automatic question answering (QA) has been an area of technical interest at TREC for more than a half decade and it has been the subject of major investment from the government-driven AQUAINT program for almost three years. QACTIS (pronounced like "cactus"), which stands for "Question-Answering for Cross-Lingual Text, Image, and Speech," is a still-in-formulation protoype system that is being developed by the U.S. Department of Defense as a means of gaining greater understanding of QA as a whole while focusing on cross-lingual and cross-media QA--areas which have received less attention from AQUAINT. The final goal for this effort is to develop a prototype which can allow users to ask questions in more than one language (e.g., English and Spanish), interpret those questions regardless of the language, and return answers which have been derived from multilingual and/or multimedia sources. Particular interest for this project is the capability of eventually answering questions from speech which is a virtually unresearched area of study. For the purposes of this TREC competition, however, we, like others, concentrate on English newswire text and the hope we have is that the knowledge gained by this experience can be useful in other languages and other media types.

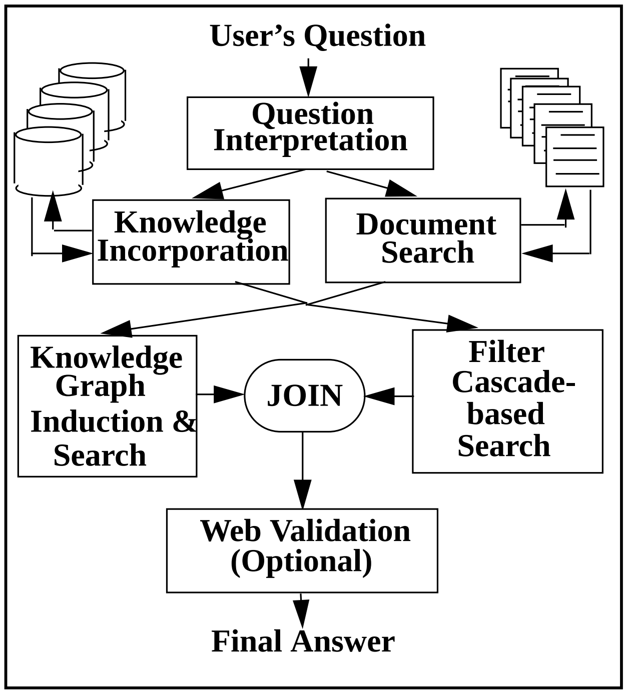
In terms of its question-answering ability, QACTIS consists of three major components. Two of these components are competing mechanisms for interpreting questions and postulating possible answers. The third component is a method of validating proposed answers. The first mechanism for answer generation is designed to be general and handle any kind of factoid-style question that might be posed. This technique, which we will here refer to as a "Knowledge-Graph Induction" strategy makes use of sophisticated natural language processing (NLP) techniques to automatically build attributed object-relationship graphs for documents which likely contain answers,. This technique then performs graphical searches to find relevant components to answers and to ensure such components have the interrelationships required by the question. This approach is fairly costly and there are many times when cheaper and more convenient solutions are available. For this reason, QACTIS has a second search mechanism which we call a "Filter Cascade" strategy. This strategy consists of finding potential answers by initially hypothesizing many possible answers and then successively applying increasingly restrictive filters to narrow down potential answers to the one or few most appropriate selections. These filters consist of template-matching, shallow grammatical rule applications, and others. The fusion of results from these two systems represents the prototype's first kind of answers. This fusion can stand on its own as a question-answerer. However, depending on a user's needs and computing environment, it might be feasible and even desirable to take advantage of the largely unstructured knowledge which exists on the World Wide Web. To satisfy this possibility, the third major component of the prototype is one which will receive answers from either of the individual systems or the fusion thereof and will use the Web to eliminate undesirable potential answers.

In the sections that follow, we provide a description of these components as well as a description of the foundations which are required in order for these components to function. Additionally, we indicate the level of performance we observed on the TREC 2004 evaluation as well as the successes and the difficulties we experienced in the evaluation. Lastly, we outline the future directions that we plan to undertake between this and the next TREC.

* Dragon Development Corporation, Columbia, MD
†Johns Hopkins Applies Physics Laboratory, Laurel MD

2. SYSTEM DESCRIPTION

Figure 1: System Overview



2.1. The Required Foundations

Figure 1 provides a depiction of QACTIS as a whole. Since this TREC evaluation focuses on the processing of English newswire text, those components that are related to multilingual or multilmedia issues have not been depicted in the diagram and we will not go into a discussion of those. Several of the components which are emphasized have been mentioned before and will be described in greater detail later: namely, the induced knowledge-graph search, the filter cascade strategy, and the web validation. Before we describe those pieces, we will first provide a description of the few remaining groundwork components, namely question interpretation, knowledge incorporation, and document search which provide the foundations to the remaining components.

2.1.1. Question Interpretation

Obviously, it is essential that before a QA system can answer a question, it must be able to receive that question from a user and provide an interpretation of that question. QACTIS can interpret questions either via batch mode or graphical user interface. Figure 2 provides a depiction of QACTIS's GUI and an example question-answer pair (where the answer also includes the corresponding document support).

Prior to the 2004 TREC QA Evaluation, QACTIS was designed to handle questions in a stateless fashion as if each question were independent of the others. Moreover, the system was primarily designed to handle exclusively factoid-style questions. Since the evaluation for 2004 was to involve question sessions consisting of factoids, lists, and definitions, these pieces has to be incorporated into the system.

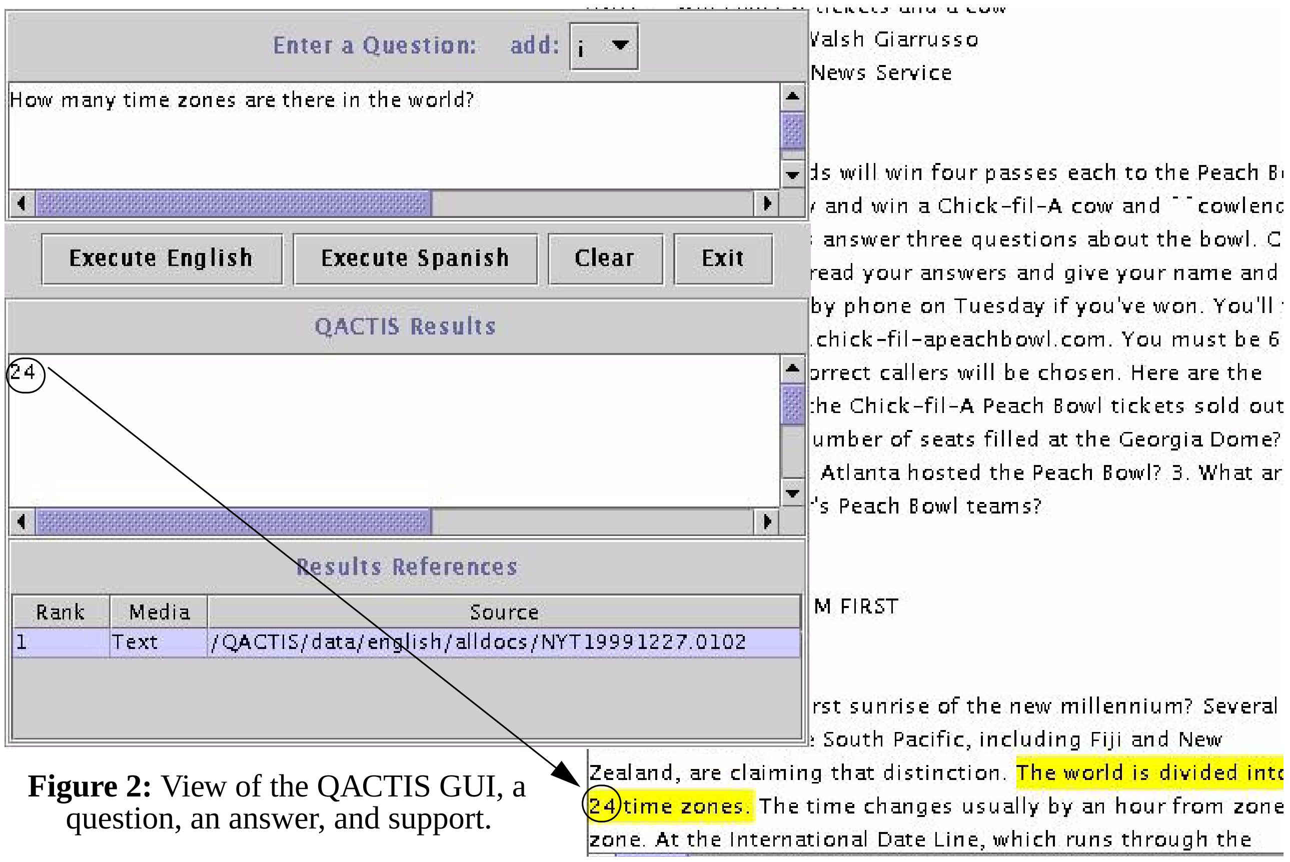
The first issue needing to be addressed was expanding from only factoids to also allow lists and 'other.' Since factoid-style answers are lists with only a single element, and since QACTIS actually produces a list and selects the best answers from that list as factoid answers, the process of deriving lists means that the system only had to be told how many answers to respond with. Based on some empirical calculations, we determined that a fixed number of list answers (between 5 and 7) seemed to provide optimal list results. These are the parameters that we then used in the evaluation.

Regarding 'other' questions, we recognized from the TREC 2003 evaluation proceedings that the best-valued definition responses were wordy and tended to report most sentences that contained words of interest [1]. We decided that our primary system would therefore select out the best sentences and return those as answers to 'other' questions. However, since older TREC evaluations had incorporated definition-like questions of the form "Who is X" or "What is X" where a factoid answer might satisfy, we thought to also try answering 'other' questions as if they were list questions of "What is X" questions. We submitted results using both methods.

The next issue that had to be tackled was to determine how to handle a series of questions within a given 'user session' (that is, under a given topic). We resolved that the easiest strategy for handling such problems would be to cast them as a list of independent questions with resolved anaphora. A source of difficulty here was to determine how complicated the TREC questions might be in terms of resolving anaphora. NIST only provided a few simple examples and it was unclear if this level of simplicity would be comparable to what would be seen at evaluation. We assumed we would need the ability to handle more complex questions. Two members of our team wrote and evaluated a large number of questions in the 'user session' format. There were six main styles of questions that we observed in this analysis. These were: (1) non-anaphoric questions; (2) questions where a simple anaphor related directly to the topic; (3) questions requiring more complicated anaphor resolution; (4) questions where neither the topic nor anaphora were indicated; (5) questions that referred to the answers to previous questions; and (6) questions that referred to the verbage of previous questions. Though the first kind of such questions is trivial to solve, the rest are more interesting. We here provide examples of each kind:

Type 2: Simple Anaphor

"Kosovo": Where is *it* located?



How many time zones are there in the world?

24

Rank	Media	Source
1	Text	/QACTIS/data/english/alldocs/NYT19991227.0102

Walsh Giarrusso
News Service

ds will win four passes each to the Peach B
and win a Chick-fil-A cow and ``cowlenc
answer three questions about the bowl. C
read your answers and give your name and
by phone on Tuesday if you've won. You'll
.chick-fil-apeachbowl.com. You must be 6
orrect callers will be chosen. Here are the
the Chick-fil-A Peach Bowl tickets sold out
umber of seats filled at the Georgia Dome?
Atlanta hosted the Peach Bowl? 3. What ar
's Peach Bowl teams?

M FIRST

rst sunrise of the new millennium? Several
South Pacific, including Fiji and New
Zealand, are claiming that distinction. The world is divided into
24 time zones. The time changes usually by an hour from zone
zone. At the International Date Line, which runs through the

Figure 2: View of the QACTIS GUI, a question, an answer, and support.

"Roses": Where were the first *ones* grown...?
"George W Bush": When did *Bush* take office?

<u>Type 3: Complex Anaphor</u>

"Synchronized Swimming": What country gave origin to *this sport*?
"Abraham & Mary Lincoln": When did *she* marry *him*?

<u>Type 4: No Anaphor nor topic in Question</u>

"Philippines":When was the Leyte Gulf invasion?
"Climate": When was the last Decadal Oscillation?

<u>Type 5: Reference to Previous Answer</u>

"Spain": Who did Spain want extradited? Where was *he* prior to extradition?
"Guerrilla": Who are the founders of ...? What is *their* nickname?

<u>Type 6: Reference to Previous Verbage</u>

"Venus": When Venus crosses the sun is known as what type of eclipse? Prior to 1999, when did the *last one* occur?

We developed software that we believed could handle the first five kinds of these questions. We anticipated that the sixth kind would be out of scope of this TREC evaluation. At evaluation time, our code was able to handle most of the questions reasonably well, but as will be made more clear in Section 3, there were some catastrophic failures which damaged our performance.

2.1.2. Knowledge Incorporation

In addition to determining the kind of question and resolving session issues, another component of question interpretation is determining the intent behind the wording of the question. Taxonomic, ontologic, and dictionary-based information can be beneficial for judging the meanings of the question words. To satisfy this need, we made appeal to versions of the English WordNet [2]. We also made use of an electronic dictionary from one of our earlier efforts [3]. In addition to accessing meaning, we also needed to contend with issues regarding syntactic variation. We made some use of WordNet's word stemming capability and we also created a stemmer of our own which would provide multiple possible stems or conflations of the words in question.

2.1.3. Document Search

A common QA practice is to initially interpret the incoming question as if it were a request for relevant documents rather than a request for an answer [1]. Using this strategy allows the system to use typical information retrieval (IR) practices to identify documents that are likely to contain answers to the user's questions. Although there is no guarantee that this assumption will be valid, this strategy has served useful in most experiements we have tried thus far. In the event that it is unsuccessful, there is not reason to believe that our QA system would have found the answer if the IR could not return appropriate documents.

We experimented with several IR systems, including one of our own making. We determined that the Lemur system [4] provided the best out-of-the-box results to our searches. We therefore incorporated Lemur into QACTIS. Although it is possible to massage the question so as to optimize IR results, we have not done this yet in QACTIS. Our results therefore reflect the use of Lemur as a black box where a query is supplied as input and a list of documents is generated as output. We use no pseudo-relevance feedback (PRF) in searching since preliminary experiments suggested that although PRF improves mean average precision, there seems to be a diluting amongst the top answer-bearing documents.

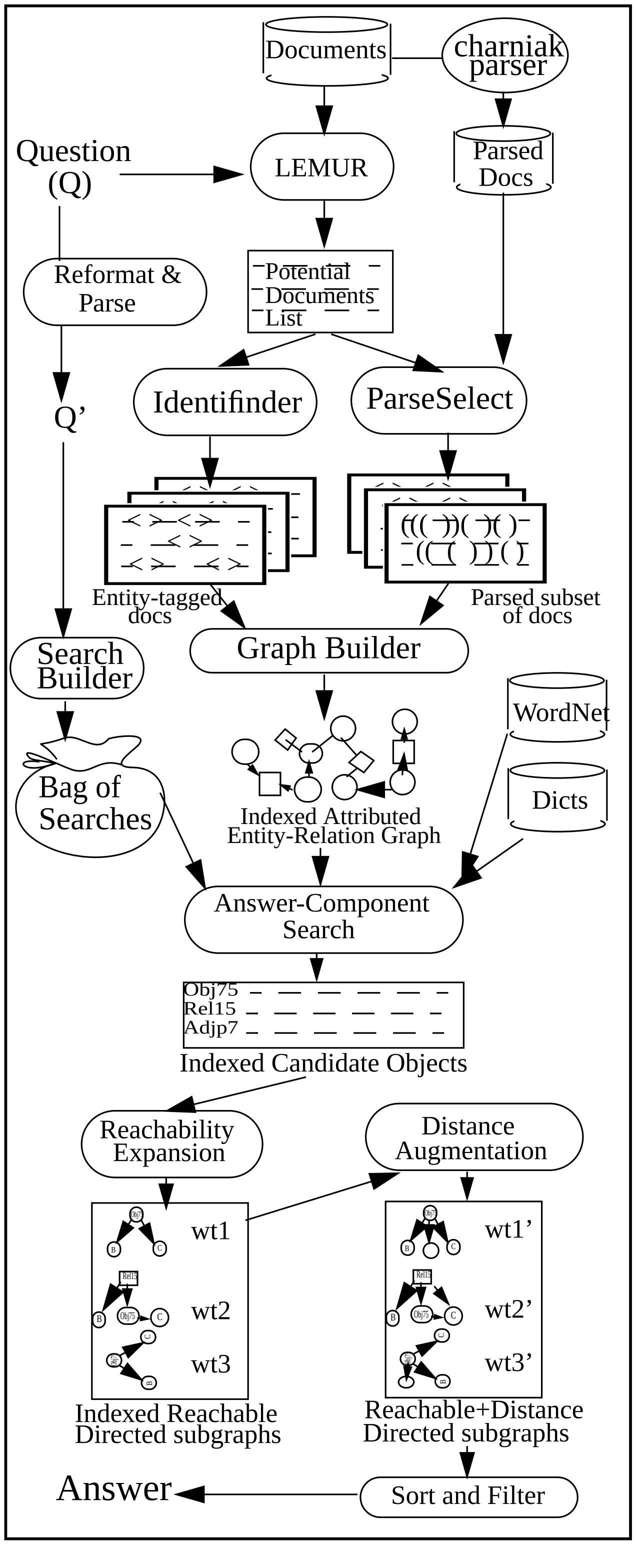
One last comment is in order regarding the kind of IR search QACTIS performs. In the QA community, some believe that the optimal form of IR for improving the ability to answer is to do optimal passage retrieval. This might be a good strategy for QA on factoid-style questions from newswire data where answers are often isolated in a limited number of sentences. However, it is not clear that restrictive passage retrieval is as beneficial when questions are more general, when answers occur across documents, and when the data consists of non-news or non-textual material. We therefore apply IR to full document retrieval which documents then get mined by the specific answer-searcher which later gets applied.

2.2. Knowledge-Graph Induction Strategy

After the introductory steps described above, the next process is to perform a search for the answer. As was mentioned before, QACTIS has a two-phase approach to question answering, namely a knowledge-graph induction strategy and a separate filter cascade strategy. The knowledge graph induction and search is to be general and to potentially handle any kind of questions whereas the filter cascade was developed to handle specific kinds of questions with greater accuracy. We first describe the general knowledge-graph strategy and later discuss the filter cascade mechanism.

Figure 3 provides a detailed graphical view of the methodology employed by the knowledge-graph induction strategy. The basic objective of this strategy is to convert the top *N* potentially relevant documents (as returned by Lemur) into a single, indexed, directed, attributed entity-relationship graph which can be mined to find connected subgraphs containing the desired components of the question. There is insufficient space to describe all of this process in exhaustive detail, so we provide a general overview of the major system components which allow us to induce and mine such a graph.

Figure 3. Knowledge-Graph Induction/Search



2.2.1. Graph Building

To begin the graph induction process, we first perform a deep syntactic parse of the Lemur documents using the Charniak parser [5]. Although we would like parsing to happen at question time, it is currently an extremely slow process (taking about 1 second per sentence on a 2.8 GHz Pentium IV). Therefore, we parse all documents earlier at indexing time and then need only to pull back the correct parses during the query phase. In addition to parsing the documents, we run BBN's Identifinder^{TM} system [6] to provide entity information which can support answering questions regarding people, places, and times. Identifinder runs sufficiently fast that we apply it at query time. Upon completion of both of these efforts, the system is then ready to induce a graph from this information.

We will consider an example of how this is done. Suppose there were a document in our collection: "Johan Vaaler invented the paper clip 90 years ago." Charniak's parser would convert this into something of the form

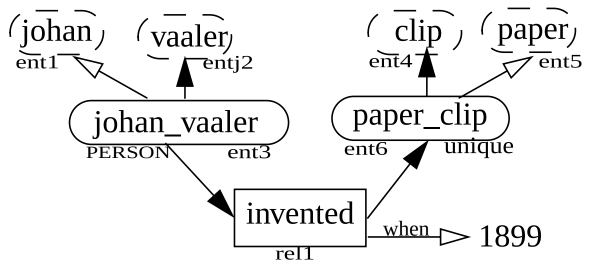
```
(S1 (S (NP (NNP Johan) (NNP Vaaler))
      (VP (VBD invented)
      (NP (DT the) (NN paper) (NN clip))
      (ADVP (NP (CD 90) (NNS years)) (RB ago))) (. .)))
```

Identifinder will also indicate that "Johan Vaaler" is a person. The graph builder then reprocesses the string to convert relative times like "90 years ago" into absolute times by using the document's metadata that indicates it was written in 1989, subtract 90 years, and reporting

```
(S1 (S (NP (NNP Johan) (NNP Vaaler))
     (VP (VBD invented)
     (NP (DT the) (NN paper) (NN clip))
     (ADVP (PP (IN in) (NN 1899)) )) (. .)))
```

The graphbuilder next converts nouns and noun phrases into entities, verb phrases into relationships, and quantifiers, prepositional phrases, and adjectives into attributes. As it does this, it tries to some degree to resolve anaphora and the meanings of definite articles. When complete, it produces a graph akin to the that in Figure 4:

Figure 4: Indexed, Attributed Entity-Rel Graph



2.2.2. Creating a Bag of Searches

In parallel with the graph-building effort, there is a separate process within the graph induction strategy which tries to interpret the user's needs based on the question. If the question were "Who invented the paper clip?," the system attempts first to convert the question into a definitive statement like "Person.Q invented the paper clip." It then parses the statement and applies the graph building process thereto. The major objects (entities, relationships, quantifiers, and attributes) and some relations between them (like quantifier and quantified) are mined from the graph as entities to search for.

Next, the system builds a set of searches to process each kind of major object. In the sentence above, the major objects would be the "person.q" and "paper clip" entities and the "invented" relationship. The system determines the kinds of objects that these are and induces a collection of subroutines (one per object) that will be used to test each word/phrase of the top *N* documents to see if each provides evidence of the object. For example, the subroutine that is induced for the word "paper clip" would test each word in the top *N* documents to see if they are WordNet synonyms of the noun "paper clip" or if they are some sort of stem or conflation of it. The same is true for the word "invented" except that it knows to be looking for synonyms of a verb. For the word "person.Q," the system knows that words or phrases that satisfy this must at least be entities, but entities that have been marked by Identifinder as people or possibly organizations provide better solutions, as might entities that have been referred to as "he," "she," and so forth. Each different kind of question word, namely "who," "when," "where," "what," "why," and so forth generate a separate type of subroutine.

2.2.3. Growth Using Reachability and Distance

The collection of subroutines is applied to all of the words/phrases within the top *N* documents and any word that is found to match a subroutine's needs is saved off as well as the location where the it appears. A language model-based score (see [7], [8]) is then used to weight each candidate answer.

The main task for this system is to determine which objects, from among those that satisfy question words, can best address the needs of the user. This means one needs to identify the objects that are candidate answers and see if, through graph connectivity, it is possible to grow a subgraph which contains all or most of the desired elements from the incoming question. Suppose Figure 4 represents one of the top *N* documents. Clearly, entity #6 ("paper clip") and relationship #1 ("invented") satisfy the question's requirement for "paper clip" and "invented." Entity #3 ("Johan Vaaler"), according to the graph, was identified as a person ... so this also satisfies the need of "person.Q."

However, these three objects by themselves do not solve the user’s question. They need to be woven together, if possible, so as to guarantee that the answer is correctly found. Since Figure 4 represents a directed graph, objects can be ‘woven together’ if an arrow exists between them and if the arrows points in the appropriate direction. From Figure 4, relationship #1 is reachable from entity #3 and entity #6 is reachable from relationship #1. Hence, we can link together each of the desired terms using this strategy and report, as a final answer, the product of the scores of each object.

For many graphs, however, it is impossible to obtain reachability between all of the needed components. As a final supplement to the answer search process, we make use of the distance that a missing component is away from a reachable subgraph. The score for incorporating such a word is related to the reciprocal of the square of its distance from the subgraph.

The scores of the distance-augmented reachable subgraphs are sorted, and the subgraphs with largest scores are reported as answers. For factoid answers, only the best such answer is kept, and for lists, the top *m* answers are reports.

2.3. Cascade of Filters Strategy

A separate module was developed to provide a more in-depth analysis and corresponding answers to certain kinds of questions such as the "How many" and “Other” type questions. This was the initial starting point for QACTIS development and the feeling was that certain types of questions are likely to be amenable to straightforward solutions. The Cascade of Filters approach (CFA) has significantly better MRR scores for questions like the “How many” over those of the Knowledge-Graph Induction algorithm, so QACTIS typically fuses both methods with the idea that specifically-tailored question answering through CFA should be used when available. The CFA uses different filters to identify potential answers and also eliminate others. Any values left at the end of all the filters was considered to be an appropriate answer.

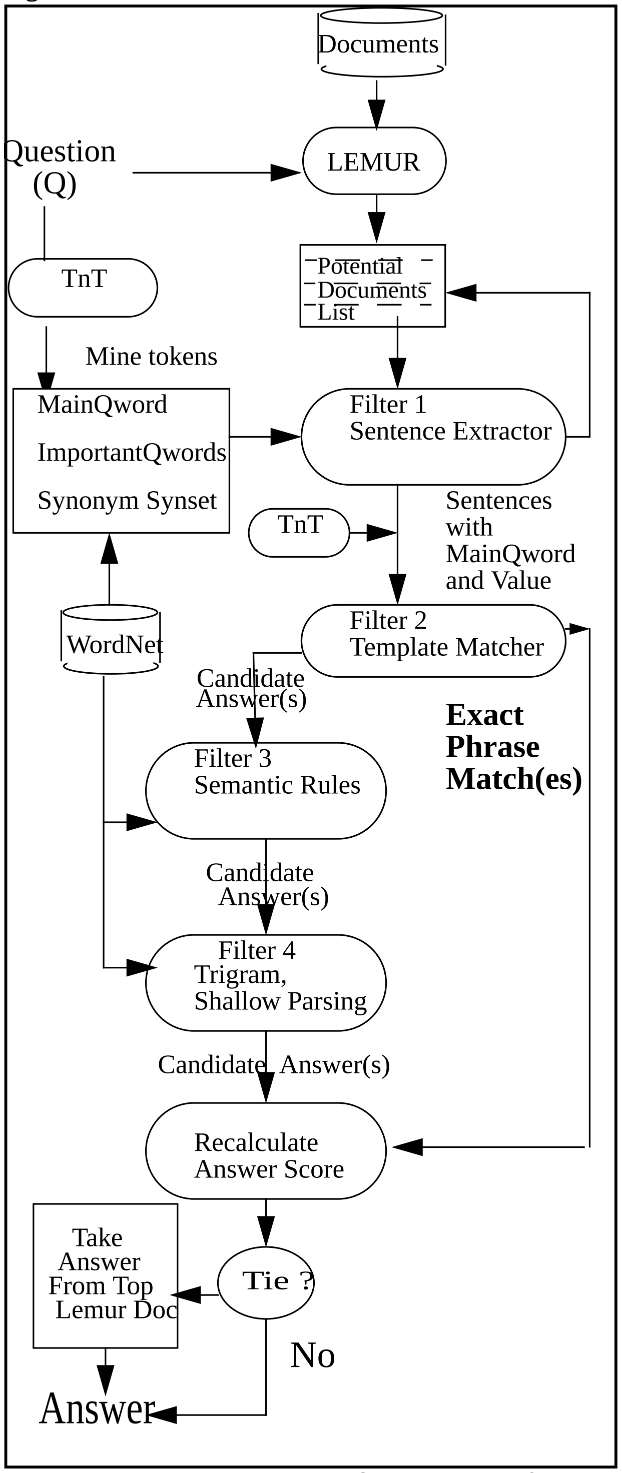
2.3.1. Trigrams n Tags - (TnT)

The CFA relies on information retrieval using Lemur, Wordnet as a lexical reference system, and TnT for part-of-speech tagging. TnT (short for “Trigrams ’n Tags”) is an efficient statistical part-of-speech tagger that is trainable on different languages and virtually any tagset [9].

2.3.2. Filters

The CFA evaluates the top *N* (*N* usually set to 30) documents returned by Lemur. (The original question is tokenized using TnT, and the main noun/noun phrase of the question is extracted.) WordNet is also used to generate synonyms for the noun/noun phrase. The retrieved

Figure 5. Cascade of Filters



documents are then successively filtered to identify candidate answers. After all candidate answers are scored, the top score is considered the correct answer and ties go to the answer from the highest scoring Lemur document. The CFA filters for “how many” questions are described below.

2.3.2.1 Sentence Extractor Filter (SEF): The SEF identifies sentences of each top IR document that contain a match to the question noun phrase or synset synonyms, and also contain a numeric value. The distance in words between the noun and value are then calculated; the shortest distance is the best. A count of the important question words and synset words is also recorded and added to the minimum distance score. Sentences with the highest scores for each of the top *N* documents are saved in a hashtable with the candidate "how many" numerical value as the key. Each of these sentences are then POS-tagged using TnT and saved in another hashtable ... again using the candidate value as the key. Both hashtables are evaluated by the Template Matcher Filter.

2.3.2.2 Template Matcher Filter (TMF): Shallow parsing was performed on the question and template match filters were formed. For example, "How many hexagons are on a soccer ball" could generate templates

a) "<#> hexagons are on a soccer ball"
b) "soccer ball has <#> hexagons"
c) "soccer ball contains <#> hexagons"

Exact matches indicated a high degree of certainty. All potential answer sentences were tested against the template filters. If any matches were found from the templates, the system progressed to the sentence score recalculation module using these template matched answers, otherwise the Semantic Rules Filter is applied to the hashtable sentences.

2.3.2.3 Semantic Rules Filter (SRF): These filters attempt to use semantic rules for validation. This set of filters eliminates candidates from the hashtable of possibilities. One such filter deals with the verb and its synonyms. For example, given the following two sentences, the first would be eliminated for the question "How many atheletes participated in the 2004 summer olympics games?"

Potential filtered sentence 1: 103 atheletes of the 2004 summer olympics won medals for the United States.

Potential filtered sentence 2: 11,099 atheletes competed for medals in the summer olympics.

Both sentences were in the candidate answer hash because they contained the main question word "athelete," had a numeric value with a distance of 1 from that question word, and contained the same number of important question words (summer olympics). In this case the verb "won" did not match "participated" or a synonym of participated (competed), so that sentence was eliminated from the candidate hashtable. The system also accounted for a conjugated form of the verb to appear in the answer as well as a verb to be presented in a different tense than the question verb. If the candidate hashtable still had entries, the processing continued to the next set of filters.

2.3.2.4 Trigram, Shallow Parsing Filter (TSPF): All word-trigrams of the question and candidate answer sentences are computed. If any candidate sentences has trigrams in common with the question, then all sentences that do not have commonality are purged. Next, the POS tags of the question are used to help form a direct object/verb/value triple from the question. Such triples are also formed for each of the remaining candidate sentences. Again, if there are any candidate answers whose triple matches that of the question, then any candidates that do not have matches are discarded.

2.3.2.5 Answer reporting: If there are still more than one candidate answer, the system rescores those that remain. The highest-scoring sentence is declared to be the answer. Ties go to the highest Lemur-scoring document.

2.4. Providing Web Validation

Several researchers have proposed using large external corpora for the purpose of validating candidate answers to factoid questions [10]. In particular, the size of the Web and the availability of online search tools make it convenient to use search engines as a data source to confirm answers using the Web as a source for validation.

Magnini et al. compared two different approaches for Web-based answer validation: a statistical approach that examined question and answer word co-occurences and a content based method that measures the proximity of question words to an answer in short text extracts. The two methods performed similarly and conferred roughly a 22% increase in performance over a baseline that did not apply answer validation. For our participation in the TREC 2004 QA task we adopted the content-based method of Magnini et al. for both the factoid and list subtasks.

The algorithm we used is based on the notion that the candidate answer pattern will appear in close proximity to the question terms. A query consisting of the exact answer pattern conjoined with content words from the question was submitted to the Altavista search engine. Thus one web request per candidate answer is required. Like many search engines, the Altavista engine returns a ranked lists of documents along with short textual extracts which contain query terms. It is these textual portions that are examined - not full documents. We want to reward answers that are very close to many question words and which are found with question words in multiple documents. We only considered the top ten responses from our baseline system which found a correct answer in 48% (TREC-9) and 37% (TREC 2003) of cases.

An answer's score is produced by aggregating scores obtained from individual textual snippets returned by the search engine. The candidate answer string can contain multiple words and may occur more than once in the extract. We search for the presence of each non-stopword question word in the snippet. If found, we determine the distance, in ‘non-question’ words, between question word and answer. (A non-question word is any word not appearing in the question). For each word, the closest location is used. A product is computed over all query terms using $2^{1/(\mathrm{dist}+1)}$ as factors. This results in a factor of 1 when a term is absent and a factor of 2 when a question word is adjacent to the answer (or separated only by other question words). Values between 1 and 2 are obtained when question words are more distant from the answer location. Per-snippet scores are summed for each candidate answer and the highest score is deemed the most likely response.

We experimented with this approach using data from the TREC-9 and TREC 2003 QA tracks and found that substantial gains could be obtained. Answer validation enhances overall performance on all factoid questions, but the technique appears more effective with certain question types. We left our baseline system’s answers unmodified for amount-type questions (i.e., "how many ..." or "how long is") which make up about 13% of the factoid questions. In Table 1 we show the improvement observed for factoid questions from previous TREC QA tracks. We produced ten candidate answers and computed both mean reciprocal rank (MRR) and the number correct (#Corr) at rank 1. Given the improvements that this technique provided at development time, we expected comparable gains to occur at evaluation time as well.

Table 1: Effectiveness of Web Validation on Previous QA Data

	Trec 9 (492)		Trec2003(380)	
	#Corr	MRR	#Corr	MRR
Baseline	101	.277	49	.190
Web-Validated	119	.310	73	.231
(Improvement)	(+18%)	(+12%)	(+49%)	(+22%)

3. SYSTEM EVALUATIONS

3.1. Description of Results

For TREC-2004, we opted to submit three separate systems. The first system (“nsaqactis1”) was our baseline system which represented the output fusion of the Knowledge-Graph and CFA strategies. CFA provided the answers to definition and “how many” style questions, and the knowledge graph produced the other types of results. The second system, “nsaqactis2,” was the same as the first except web validation was applied afterwards. The third system, “nsaqactis3” used only the knowledge graph and web validation to answer the questions. The performance of the TREC 2004 evaluation is provided in Table 2. To our surprise, our baseline (#1) system performed approximately as well as our Trec 9 baseline had performed. We had hoped that a comparable web-validation improvement would have been seen as well, but to our dismay, web validation actually resulted in a loss of performance on non-list questions, but there was a reasonable gain using web validation on lists. We were also delighted to have “Other” style questions perform very well given that this was a last-minute and untested feature of our system. In the sections that following, we provide a brief analysis of these results to include an indication of where things went as well or better than expected and where the system failed to respond as desired.

Table 2: TREC 2004 Performance

Strategy	Factoid	List	Other	All
Knowledge Graph+ Filter Cascade (#1)	0.204	0.071	0.367	**0.211**
System #1+ Web Validation (#2)	0.187	0.098	0.355	0.207
Knowledge Graph+ Web Validation (#3)	0.183	0.104	0.062	0.133

3.2. Times When Things Go Well

Perhaps the component of our system whose performance we are most pleased with is that of “Other” answering. Early in the paper, we mentioned that just prior to the TREC evaluation, QACTIS had very little capacity to handle “other” style questions. Through the Knowledge-Induction Search we did have limited capacity of declaring hypernyms of the entities needing definition to be defining statements about those entities, but we did not have the ability to handle definitions in the encyclopedic style that TREC evaluates. For interest sake, we did submit our hypernym output as one version (#3) of definitions, but we had interest in creating definitions that would be more aligned with the evaluation. In last year’s TREC evaluations, definitions that consisted of well-chosen sentences seemed to outperform most other types of definitions, so our throught was to build some comparable mechanism. The CFA strategy works primarily from well-chosen sentences, so it seems well-suited to building up our definer. Definitional questions were therefore handled by modifying the Sentence Extractor Filter (2.3.2.1) used in CFA. In particular, sentences from the top *N* documents were scanned for matches to the target topic and those sentences were saved off. No attempt was made to avoid redundancy with prior questions from the same session. A simple filter was then applied that accepted only those sentences as answer components that had lengths greater than 10 and less than 250 words. The number of sentences per topic was capped at 50.

Table 2 suggests that this strategy seemed to work reasonably well. In our best-scoring run (the baseline, #1), the answers were on average about 2336 characters long and had an average precision of 0.179 and an average recall of 0.485. Since recall played a significant role in performance, and since long answers are well tolerated, this led to the reasonable score of 0.367 which was better than we had hoped to achieve.

(As a matter of comment, the third system, which scored poorly was based purely on lists of potential hypernyms. In this case, the average answer length was reduced to 226 characters and the precision dropped somewhat to 0.118. Yet the average recall -- the most important component of the evaluation -- was very low...only 0.066.)

3.3. Difficulties: The Unexpected

3.3.1. Series-style Question Problems

We had expected that our module to convert series-style questions into independent questions would work satisfactorily. There were a number of questions where the system had significant difficulties, however.

In the 11th squestion-session, the target was “the band Nirvana” and the second and third questions of that session asked about the band members and the band’s formation. The resolver recognized that “band” and “band Nirvana” were the same but did not make the corresponding substitutions into the independent questions. Thus, the question did not indicate what kind of band was being looked for and all corresponding answers were missed.

In this same series, the next questions that were posed used the terms “their” and “they.” The resolver assumed that since the former question mentioned band members whereas the topical ‘band’ is a singular entity, “their” and “they” must refer to the answer of the second question in the series. Thus, instead of using “Nirvana” as a substitution for the anaphora, it used what it thought mught be an answer to the group members question, and it thus substituted “Desmond Chase” for “they” and “their.”

These two phenomena, namely, failing to make the substitution when part of the topic was found in the question *and* using answers to previous series questions to resolve the anaphors, was a condition which occurred a total of 16 times throughout the question collection.

3.3.2. Inexact answers

Another difficulty for our system was that by and large, until the evaluation, we paid little heed to the actual documents where answers appeared and we did not concern ourselves heavily on whether the cleanest answer (i.e., an exact answer) was presented -- but only if the right kind of answer was identified. The evaluation, however, was concerned with providing document support and answer exactness.

The first of these issues is rather complicated to resolve in a system which tries to identify answers across documents -- which document did the answer actually appear in? To prepare for the evaluation, we had to equip the system to know, to the best of its ability, where the original answer came from. These modifications were reasonably successful in that only one of the otherwise correct answers we returned was declared to be unsupported.

On the other hand, the requirement for answer exactness resulted in a high penalty for us. Our best system only produced 47 correct answers. Yet the system produced 12 answers which were deemed to be inexact, thus resulting in significant damage to our overall performance.

3.3.3. Experiments and Test in Web Validation

As mentioned, we applied answer validation on two of three TREC 2004 submissions. Although the method was untested, we also attempted to validate responses to list questions as with factoids.

Table 3: Comparing performance by question type

	Total	w/o Validation	w/ Validation
Abbrev	1	0	0
Amount	31	7	7
Location	28	8	4
Miscellany	15	1	1
Process	2	0	0
Time	48	19	19
What	66	6	7
Who	39	6	5
	230	47	43

Location questions fared worse with Web-validation and ’what’ and ’who’ questions were less popular in this year’s set of questions. These differences in the distribution of question types might account for the observed differences.

For list questions we returned as many as 7 responses after considering up to 15 possibilities from the baseline system (more or less possibilities may have been provided). Here Web-validation had a significant positive effect, improving the F-score from 0.071 (without validation) to 0.098.

4 FUTURE DIRECTIONS

The results we have presented represent our first attempt to participate in the TREC QA Evaluation. Across the course of the next year, we expect that our system will improve as we add functionality for better resolving anaphora, so this will be one of early expansions. Moreover, as we have analyzed documents, we have noted that a significant number of potential answers are not observed since inference is required in order to find them. This, then, will be our next area of concern. Lastly, since multimedia and multilingual QA is of interest to us, we expect to enable the capabilities we have shown here in Spanish and began to try them on non-text media.

5 ACKNOWLEDGMENTS

The authors would like to thank Carol Vaness-Dykema for her efforts to help establish and oversee the foundations of the QACTIS effort. Additional appreciation goes to John Prange and the AQUAINT program for providing funding for portions of the work involved in this effort.

6 REFERENCES

[1] E.M. Vorhees. “Overview of TREC 2003.” National Institute of Standards and Technology, Gaithersburg, MD, 2003.

[2] Miller G. A., Beckwith R., Fellbaum C., Gross D., and Miller K. J. “WordNet: An online lexical database.” *International Journal of Lexicography* 3(4): 235-244, 1990.

[3] P. Schone, J. Townsend, C. Olano, T.H. Crystal. “Text Retrieval via Semantic Forests.” TREC-6, Gaithersburg, MD. *NIST Special Publication 500-240*, pp. 761-773, 1997

[4] The LEMUR System. URL: http://www-2.cs.cmu.edu/~lemur

[5] E. Charniak. “A maximum-entropy inspired parser.” *Technical Report CS-99-12*, Brown University, 1999.

[6] D. Bikel, R. Schwartz, R. Weischedel, "An Algorithm that Learns What’s in a Name," *Machine Learning*, 1999

[7] P. McNamee, J. Mayfield. “Character N-Gram Tokenization for European Language Text Retrieval: Special Issue on CLEF.” *Information Retrieval*, vol. 7, no. 1-2, pp. 73-97(25), 2004.

[8] J.M. Ponte, W.B. Croft. “A language modeling approach to information retrieval.” *Proc. of 21st Annual Int’l ACM SIGIR Conference of Research and Development in Information Retrieval*, Melbourne, Australia, pp. 275-281.

[9] T. Brants, “TnT - a statistical part-of-speech tagger.” *Proc. of the Sixth Applied Natural Language Processing Conference ANLP*, 2000.

[10] B. Magnini, M. Negri, R. Prevete, and H. Tanev, ’Comparing Statistical and Content-Based Techniques for Answer Validation on the Web.’ In *Proceedings of the VIII Convegno AI*IA*, Siena - Italy, September 11-13, 2002.