# A Multi-Strategy and Multi-Source Approach to Question Answering

**Jennifer Chu-Carroll** **John Prager** **Christopher Welty**
**Krzysztof Czuba** **David Ferrucci**
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598, USA
jencc,jprager,welty,kczuba,ferrucci@us.ibm.com

## 1 Introduction

Traditional question answering systems typically employ a single pipeline architecture, consisting roughly of three components: question analysis, search, and answer selection (see e.g., (Clarke et al., 2001a; Hovy et al., 2000; Moldovan et al., 2000; Prager et al., 2000)). The knowledge sources utilized by these systems to date primarily focus on the corpus from which answers are to be retrieved, WordNet, and the Web (see e.g., (Clarke et al., 2001b; Pasca and Harabagiu, 2001; Prager et al., 2001)). More recent research has shown that introducing feedback loops into the traditional pipeline architecture results in a performance gain (Harabagiu et al., 2001).

We are interested in improving the performance of QA systems by breaking away from the strict pipeline architecture. In addition, we require an architecture that allows for *hybridization* at low development cost and facilitates experimentation with different instantiations of system components. Our resulting architecture is one that is modular and easily extensible, and allows for multiple *answering agents* to address the same question in parallel and for their results to be combined.

Our new question answering system, PIQUANT, adopts this flexible architecture. The answering agents currently implemented in PIQUANT vary both in terms of the strategies used and the knowledge sources consulted. For example, an answering agent may employ statistical methods for extracting answers to questions from a large corpus, while another answering agent may transform select natural language questions into logical forms and query structured knowledge sources for answers.

In this paper, we first describe the architecture on which PIQUANT is based. We then describe the answering agents currently implemented within the PIQUANT system, and how they were configured for our TREC2002 runs. Finally, we show that significant performance improvement was achieved by our multi-agent architecture by comparing our TREC2002 results against individual answering agent performance.

## 2 A Modular and Extensible QA Architecture

The architecture adopted by our PIQUANT system, shown in Figure 1, defines several basic roles that components of a QA system can play. The definition of each role includes a consistent interface that allows components implementing that role to be easily plugged into the system. This architectural approach is not simply to facilitate good software engineering in a group, but it allows hybridization at a fairly low development cost, and it also facilitates experimentation based on the choices available within the different component roles.

The main components of our architecture are briefly described as follows:

1. **Question Analysis** components analyze questions to produce information consumed by other components in the form of a QFrame. Information contained in the QFrame should minimally include a question type that would help guide the selection of one or more answering agents (see below) appropriate for addressing the question. A QA system typically has one question analysis component, but may possibly have as many as one per answering agent.

2. **Answering Agent** components implement answer finding strategies given the results of question analysis and a knowledge source. These may be as simple as composing a bag-of-words query for document/passage retrieval, or as complex as breaking the question into sub-questions and consulting multiple knowledge sources. We expect QA systems to have multiple answering agents that pursue different strategies in parallel, which we believe to be an important feature of our architecture: not only can we experiment with different question answering strategies and knowledge sources, but with combining them as well.

3. **Answer Resolution** components combine the results of multiple answering agents into a single rank-

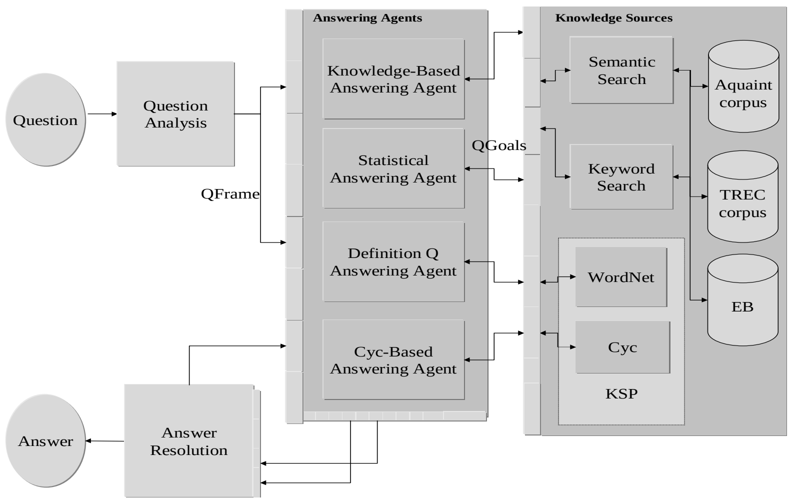

Figure 1: PIQUANT's Architecture

ing. These components may simply perform ranking over the combined set of answers of all answering agents, or may do something more complex such as feeding answers from one agent back into others. Ultimately, the final answers of a QA system are provided by this component, so there is only one such component in any QA system.

4. **Knowledge Source Adapter** components insulate the other components of the QA system that consult knowledge sources from the multitude of data formats, access mechanisms, representation languages, reasoning services, and ontologies that consumers of existing structured knowledge sources must be acutely aware of.

An obvious benefit of our component-based approach is that we can easily experiment with and compare different techniques for filling these roles by keeping the rest of the components of the QA system fixed and changing only the components that implement the techniques we wish to compare. Thus we could, for example, measure the overall impact on QA performance of using statistical vs. rule-based annotators, or using machine learning vs. rule-based answering agents. In addition, as noted above, we can often *combine* the strengths of different techniques to improve overall performance, which will be the focus of this paper.

## 3 A Multi-Agent Approach to Question Answering

Answering agents that can be adopted for QA may differ along various dimensions. One such dimension is the type of knowledge source from which answers are extracted, which may include unstructured text resources or structured knowledge sources such as Cyc (Lenat, 1995) or WordNet (Miller, 1995). Even when two answering agents consult the same knowledge source, they may adopt different processing strategies. For example, existing question answering systems vary greatly, from utilizing primarily knowledge-driven components, e.g., (Harabagiu et al., 2001; Prager et al., 2000) to adopting mainly statistical methods, e.g., (Ittycheriah et al., 2001; Ravichandran and Hovy, 2002).

We have so far integrated into our PIQUANT system answering agents that utilize both structured and unstructured knowledge sources. For the latter class, we have incorporated two answering agents adopting fundamentally different processing strategies. This section describes each of these answering agents, as well as how their answers are combined to formulate the system's final answers.

### 3.1 Agents Based on Unstructured Information

Perhaps motivated by the TREC QA track, the vast majority of existing question answering systems adopt a large text corpus as their information source. Additionally, while many such systems adopt a classic pipeline architecture, each typically employs a different approach in instantiating its components. Currently, we have incorporated two text-based answering agents into PIQUANT, one utilizing a primarily knowledge-driven approach and the other adopting statistical methods. These two answering agents have performed quite comparably in past TREC QA tracks.

### 3.1.1 Knowledge-Based Answering Agent

Our first answering agent utilizes a primarily knowledge-driven approach to question answering, based on Predictive Annotation (Prager et al., 2000; Prager et al., 2003). A key characteristic of this system is that potential answers, such as person names, locations, and dates, in the text corpus are *predictively annotated*. In other words, the text corpus is indexed not only with keywords, as is typical for most search engines, but also with the semantic classes of these pre-identified potential answers.

During the question analysis phase, a rule-based mechanism is used to determine one or more of about 80 semantic types of the candidate answer, along with a set of keywords. A weighted search engine query is then constructed from the keywords and the candidate semantic classes. The search engine then returns a small (typically 10-passage) set of 1-to-3-sentence passages based on the query. The candidate answers in these passages are identified and ranked based on three criteria: 1) match in semantic type between candidate answer and expected answer, 2) match in weighted grammatical relationships between question and answer passages, and 3) answer frequency.

### 3.1.2 Statistical Answering Agent

The second answering agent used in PIQUANT is the statistical question answering system of Ittycheriah et al. (Ittycheriah et al., 2001). This statistical answering agent is also based on the pipeline architecture; however, instead of adopting rule-based mechanisms, it utilizes a maximum entropy approach for training system components.

In question analysis, one of a set of 32 potential answer types is selected based on features such as words, POS tags, bigrams, and question word markers. The search module adopts a two-pass approach in which high scoring passages from an encyclopedia are used to augment the query terms, which are then used for search against the TREC corpus. The search engine returns a large set of passages (100) for further consideration. Named entities and their semantic types are identified from these passages, again using a maximum entropy based mechanism, and a confidence value computed for each named entity based on its likelihood of being a correct answer to the given question.

## 3.2 Agents Based on Structured Knowledge Sources

It has been previously established that finding the answers to questions in structured knowledge sources such as WordNet and including the answer in a bag of words can improve accuracy (Prager et al., 2001). We have expanded on this notion in two ways, first by adapting a wide variety of knowledge sources into our QA system, and second by handling the case of numerical answers in post-hoc answer filtering.

### 3.2.1 Knowledge Server Portal

For certain classes of routine fact-seeking questions, such as populations and capitals of geo-political entities, the answering agent recognizes a number of ways of asking these questions and formulates a query to a *structured knowledge source.* These knowledge sources include public databases such as the US Geological Survey, websites with data in formatted tables from websites such as `http://www.UselessKnowledge.com`, public domain lexicons such as WordNet, and the Cyc knowledge base.

Each of these knowledge sources is maintained by external groups and is out of our direct control. Each source has data in a different format, requires a different access mechanism, is expressed in a different representation language, provides different reasoning services, and assumes a different ontology. In addition, this external control means that any of these formats, access mechanisms, etc., may change, and of course adding new knowledge sources introduces a new set of choices to be aware of.

Rather than require that each answering agent understand all these dependencies in order to use the knowledge sources, we have isolated the role of adapting external structured knowledge sources and presenting a consistent set of choices to all the QA components through a set of knowledge-source adapters. We refer to the system component that provides access to these knowledge-source adapters as the *knowledge server portal (KSP)*. The adapters provided by KSP support the set of queries the question analysis component is capable of recognizing, such as “What is the capital of Syria?” or “What is the state bird of Alaska?”, and are responsible for composing the proper query to the knowledge sources that may have the answer. The answering agent then may formulate a query that includes the answer as a search term similar to (Prager et al., 2001).

### 3.2.2 Cyc Sanity Checker

For certain questions, in particular questions that have numerical answers, adding the answer as a search term is not effective, because there are innumerable variations on the way the number may be expressed in the corpus. Populations, for example, vary over time by a significant amount, and are usually in the millions. For a question like, “What is the population of Maryland?”, knowing that the latest figure for the population of Maryland is 5,296,486 does not quite help us search the corpus, because we are almost guaranteed that precise number will not appear. It could be expressed as “5 million”, “5.1 million”, “5.3 million”, or “5,200,390”, etc. This process is complicated further when unit conversions are required, as in the question, “How big is Australia?” In addition to having to find a number in the vicinity of “1 million square miles”, we also need to account for the fact that the passage may talk about square kilometers, or acres. Instead of folding the known answer into the query in cases like this, we allow the question answering system’s regular procedure to generate a set of candidate answers first, and check them to be within some experimentally determined range of the answer the knowledge source provides.

We have implemented the validation of answers with numerical values using an interface to Cyc called the Cyc sanity checker. The sanity checker is invoked with the expected semantic type of the answer (such as POPULATION in the first example above), the focus of the question (“Maryland”), and the system’s proposed answer (“X people”). It returns one of the following verdicts: “in range”, if the proposed answer is within a certain “fudge factor” (currently 10%) of the value in Cyc’s knowledge base, “out of range”, if the value falls outside of the acceptable range of values, or “don’t know”, indicating that Cyc either has no information about the focus itself, or about the particular attribute in question about the focus.

### 3.3 Answer Resolution — Putting it All Together

We have described four independent answering agents currently incorporated into our multi-agent architecture. With the exception of the Cyc sanity checker, which is invoked as a post-hoc filtering process for rejecting unreasonable answers, the other three answering agents actively contribute potential answers to a given question. It is then the task of the answer resolution component to determine how the various answers proposed by each answering agent should be combined and reconciled.

Because of the TREC requirement that all answers be justified by passages from the given corpus (henceforth referred to as the AQUAINT corpus), we feed potential answers given by KSP back into the search process to identify relevant passages in a process similar to that described in (Prager et al., 2001). These passages typically contain answers identified by KSP, as well as relevant question terms; thus, they are good candidate passages for locating justification for the answer provided by KSP in the reference corpus. Because of this answer feedback mechanism, all answering agents produce relevant passages and ranked candidate answers in a uniform fashion, simplifying the answer resolution process.

Currently, PIQUANT’s answer resolution component allows for merging at two different points in the pipeline as follows:

- Passages proposed by multiple answering agents can be combined to feed through the answer selection component of our knowledge-based answering agent.
- Candidate answers proposed by different answering agents can contribute to determining PIQUANT’s final output.

In addition to determining the answer to a given question, the answer resolution also computes a *confidence value* indicating the system’s certainty in the given answer being a correct answer to the question. This confidence value can then be used for ranking system responses for TREC submissions.

## 4 Recognizing When the System Does Not Know

To make the task more realistic, the test set for the QA track contains a number of questions for which no answer can be found in the document collection, as verified by NIST (we call such questions “NIL questions” or “no-answer questions”). To simplify the task of detecting no-answer questions, we reduce it to the problem of finding the questions for which we can reasonably assume that the system was not able to find a correct answer. This is a much weaker condition since it is dependent on the answer search strategy the system implements, i.e., there might be other strategies that would be successful at finding an answer. It can, however, be implemented easily by setting a threshold on the confidence value that is assigned to a question by the answer resolution module.

We implemented two strategies for determining which questions had no answers: a knowledge-based strategy and strategy based on confidence processing. The knowledge-based strategy makes use of KSP and is evoked for questions that were classified as appropriate for KSP look-up. If KSP was able to provide an answer to such a question and the answer string could not be found in the collection, we assumed with high confidence that the question is a NIL-question. Since KSP has only recently been integrated into the system and the number of questions that are referred to it is still limited, this NIL-assignment strategy applied to only two questions in the final submission.

In our confidence-based approach, we adopted a two-stage processing strategy for detecting and ranking no-answer questions. The first stage detects which questions are likely to have no answer in the collection by comparing their scores with a trained confidence threshold. The second stage takes care of the proper ranking of questions likely to have no answers by increasing their rank.

In order to train the NIL assignment algorithm, we ran our system on the TREC-10 question set and plotted the distribution of different question types in the final ranking. We marked the questions that did not have an answer according to NIST, the questions for which the system produced a correct answer, and the questions for which the system’s output was wrong. The resulting plot is in

```
                                                    NIL  CORRECT
         xxxxxxxxxxxxxx.xx.xxxxxxxxxxxxxx.x..xx.xx   0      35
xxxxxx-x.-x.xxxxxxxx..x-xxxxxxxxxx.xxxxx.x.xxx.-xx   4      38
xx.....x.-xx.....xx....x.xx.x..xxx.xx...xx.x..xx.x   1      22
.-...x.xx-..x..x.xx....xx.x...xx.....x..xxx....xx.   2      18
........x....x..xxxx...x...xx....xxxxx--......xxx.   2      17
..x.xxx...-x-...xx.....x...xx--.xx-....xx..x..x...   5      16
..x.x.-......x....x.x-.x.xx...-x-x-x-...-..x-x.x.x   8      15
x..-x.....x.x.....-..........-...-..x.-....-..x...   6       6
.x--......xx....-.-..x.-....-.-..x...........--...   9       5
-.-.-..--...-x.xx....-.-x......-.....-.-...-.x.-.   13       5

x correctly answered question
. incorrectly answered question
- NIL question according to NIST
```

Figure 2: TREC-10 training data for NIL assignment

Figure 2. It represents the 491 Trec-10 questions (9 questions were thrown out, see (Voorhees, 2001)) split into blocks of 50 (based on TREC-10 we expected approximately 50, or 10%, of the questions to have no answers). Next to each block we plotted the number of questions within that block that were answered correctly and the number of NIL questions within that block. As can be seen in Figure 2, the numbers change almost monotonically, which suggests that the confidences produced by the system could be a reasonably reliable indicator of the system's performance on a given question.

According to Figure 2, the final two blocks contain more NIL questions than correctly answered questions. This means that changing the system's answer to NIL for all the questions in these two blocks will produce a net gain of 12 correctly answered questions. It will also change the incorrect answers to NIL for 68 questions, which is valuable from the user's point of view, assuming that "NIL" could be interpreted as "I don't know."[1]

Based on this analysis we manually picked (on the training data) a confidence threshold that would allow us to select the 100 lowest ranked questions. We used the same threshold on the test set and changed whatever answers the system found for the questions below the threshold to NIL.

We also looked at how the average precision changed within a 50-question window as we moved it by one question at a time down the ranking, and we found the trend to be close to monotonic. Since changing the answers in the final two blocks to NIL caused the average precision within these blocks to increase, we decided to move the two blocks higher in the ranking to the rank with the same average precision. We computed the difference in confidence value between the answer at the target rank and the highest ranking answer in the NIL-block. This difference was then added to the confidence values of all NIL answers in our runs submitted to TREC.

[1]If the systems participating in the competition were penalized for providing incorrect answers, the questions in the third-to-last block could also be changed to NIL with no net gain in the number of correctly answered questions but significantly fewer potentially confusing answers.

## 5 Performance Evaluation

### 5.1 Experimental Setup

For the 2002 TREC QA track, we submitted three runs, each evaluating a different aspect of PIQUANT's multi-strategy, multi-source architecture. These three runs were set up as follows:

1. Run "IBMPQ" exploits the multi-source aspect of PIQUANT with the knowledge-based answering agent. However, instead of only searching in the AQUAINT corpus for relevant passages, we adopt two other supporting corpora: the corpus used in the TRECs 8-10 QA tracks (henceforth referred to as the TREC corpus) and a subset of the Encyclopedia Britannica. A corpus plays a supporting role when candidate answers found in that corpus can be used to boost the confidence of the same answer found in the main corpus, but the corpus cannot propose new answers not found in the main corpus.

2. Run "IBMPQSQA" exploits the multi-strategy aspect of PIQUANT by incorporating results from the SQA statistical answering agent made available to us by Ittycheriah and Roukos (Ittycheriah et al., 2001). The knowledge-based answering agent was configured to retrieve relevant passages from the AQUAINT and TREC corpora. Additionally, the top 10 passages with the correct answer type retrieved by the statistical answering agent were also considered. PIQUANT's answer resolution component then selects and ranks answers based on passages from the three answering agents/sources. Once the top answer for each question is determined, PIQUANT's confidence score for the answer is adjusted based on the answer given independently by the statistical answering agent. A large boost in confidence is given to identical answers proposed by both systems, whereas a small boost in confidence is given to partially overlapping answers.

3. Run "IBMPQSQACYC" examines the effect of the Cyc sanity checker as a post-hoc filtering process. The system is configured exactly as in run "IBMPQSQA" with the following exception. Prior to determining the top answer for each question, PIQUANT repeatedly invokes the Cyc sanity checker with a semantic representation of the question and the topmost uneliminated candidate answer as long as the sanity checker deems the given answer "out of range". PIQUANT then eventually selects its most confident answer acceptable to the sanity checker. Note that if this top ranked answer is considered "in

range" (as opposed to "don't know"), its confidence is given a strong boost, as it is independently validated by a structured knowledge source.

After PIQUANT generates the answer to each question and its associated confidence, the NIL-assignment process discussed in Section 4 is invoked. As a result, answers with low confidences were changed to NIL and their confidences slightly increased.

### 5.2 Results and Analysis

#### 5.2.1 Results of Submitted Runs

Table 1 shows the results of our three runs both in terms of percent correct and average precision. For comparison purposes, it shows, in addition, the performance of the statistical answering agent submitted independently to the same track (ibmsqa02a) (Ittycheriah and Roukos, 2002), as well as the performance of the knowledge-based answering agent using only the AQUAINT corpus (PQ single).[2] A comparison between the results for PQ single and IBMPQ shows the impact of the multi-source aspect of PIQUANT. Our results show that by attempting to identify supporting evidence from two additional corpora, the system achieved 19.9% relative improvement in the percentage of correct answers, and the average precision score improved by 14.6%. A comparison of the results for runs IBMPQ, ibmsqa02a, and IBMPSQA shows the contribution of adopting multiple strategies for question answering in PIQUANT. Although the percentage of questions answered correctly improved for both systems (from 33.8% for IBMPQ and 28% for ibmsqa02a to 35.6% combined), the gain in average precision is much more substantial (9.7% relative improvement compared to IBMPQ). This confirms our intuition that when answering agents (semi-)independently arrive at the same answer, we can be more confident that the answer is a correct one. A comparison of the results for runs IBMPQSQA and IBMPQSQACYC illustrates the impact of the Cyc sanity checker. Although the impact as shown is very minimal, we should note that because of the limitations in PIQUANT's current question understanding capabilities, the sanity checker was invoked only for 3 out of the 500 questions (although there were several more questions which fit the profile but were not detected as such). Additionally, out of the 3 questions, Cyc only had knowledge about one of them, *"What is the population of Maryland?"* It is the effect of sanity checking on this question that led to the improved performance for our last run. PIQUANT's top ranked answer for this question in run IBMPQSQA was "50,000", from the sentence "Maryland's population is 50,000 and growing rapidly." This would otherwise be an excellent answer if it were not for the fact that the article from which this passage is extracted discusses (the Maryland population of) an exotic species called nutria. By employing sanity checking, however, PIQUANT was able to consider that answer "out of range", and return an initially lower-ranked correct answer "5.1 million" instead with high confidence.

[2]The results for PQ single were obtained by manual evaluation by one of the authors with reference to available judgments by NIST accessors and answer patterns made available by Ken Litkowski.

#### 5.2.2 Effects of NIL Assignment

In our best submission run (IBMPQSQACYC), the confidence-based NIL-assignment strategy resulted in 147 NIL answers, which was more than we anticipated. This is due to the generally lower confidences on a new question set. The system correctly assigned NIL to 29 out of 46 questions, which translates to a recall of 0.630 and precision of 0.196. By assigning the NIL answers, the system changed 9 correct answers incorrectly to NIL, which gave us a net gain of 20 questions (given the answer pattern set currently available to us). The questions for which the answer was changed to NIL were then moved to rank 288, which resulted in a very minimal (below 0.5%) improvement in the final average precision score.

#### 5.2.3 Analysis of the Average Precision Metric

If the same scoring method had been used this year as in previous TREC QA tracks, the mean reciprocal rank (MRR), exercised over a single answer per question would amount to a simple count of number correct. However, in order to begin to tackle the issue of answer reliability, answers this year were returned by participants in decreasing order of system confidence (although no numerical values representing confidence were returned). The systems' final scores were evaluated by Average Precision, the average being computed over the first answer, the first two answers, the first three answers, and so on up to the whole set. Clearly, this gives considerably more relative weight to the earlier answers, and considerably less to the last answers. The contribution $c_k$ of a correct answer in position $k$ out of $N$ questions in total is given by $ln(\frac{N+1}{k+1}) \leq Nc_k \leq ln(\frac{N}{k}) + \frac{1}{k}$.

The plot in Figure 3 shows this contribution, in units of 1/500, for positions 1 to 500 for a set of 500 questions. Relative to a score of approximately 1 unit for the greater part of the range, the contribution of the first position is nearly 7, indicating how important it is for systems to sort their submissions well.

Another view of the evaluation space introduced by the Average Precision metric is presented in Figure 4. The diagonal line and "cloud" represent what happens with no attempt to sort the results. The solid line in the center of the cloud is the ideally-uniformly-distributed case (i.e. if 1/3 of the answers are right, the submitted list goes ...RWWRWWRWWRWW...), and the

|  | IBMPQ | IBMPQSQA | IBMPQSQACYC | ibmsqa02a | PQ single |
|---|---|---|---|---|---|
| % Correct | 33.8% | 35.6% | 35.8% | 28.0% | 28.2% |
| Avg Prec | 0.534 | 0.586 | 0.588 | 0.454 | 0.466 |

Table 1: PIQUANT's TREC 2002 Run Results

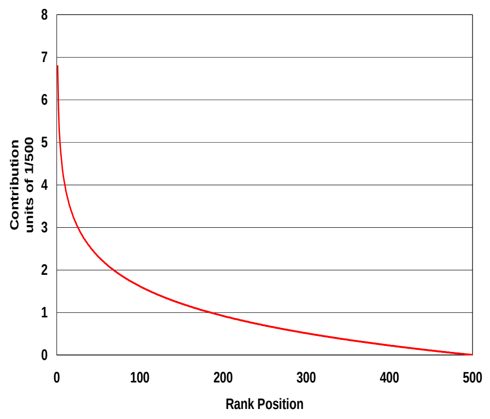

Figure 3: Contribution of Correct Answers to Average Precision Score

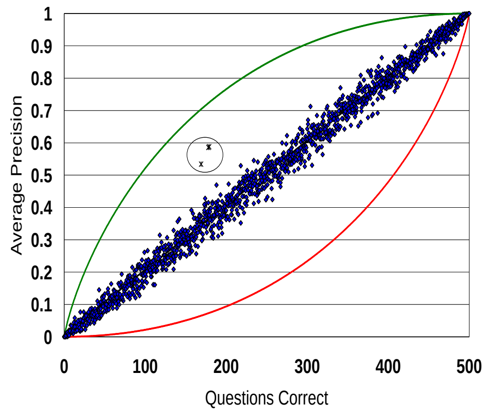

Figure 4: Upperbounds and Lowerbounds for Average Precision Scores

cloud is a simulation of randomly-distributed rights and wrongs, given the number of correct answers. The width of the cloud approximately represents a 3-standard-deviation spread. The upper curve is the optimal case (e.g. RRRRRR......WWWWWW), while the lower curve is the pessimal case (i.e. all the right answers are sorted to the end.)

### 5.2.4 Ranking Ability

The circled points in the middle of Figure 4 represent our TREC runs. The maximum possible score, represented by the upper curve, for $n$ correct out of $N$ is approximately $\frac{n}{N}(1 + ln\frac{N}{n+1})$. By examining how far up a virtual vertical line from the diagonal (*expected*) to the upper curve (*max*) a plotted point (*actual*) lies, one can see how well the system sorted its answers for submission - i.e. how well it knows what it knows. This fraction, which we call the *Ranking Ability*, can be computed as $\frac{actual-expected}{max-expected}$. In the case of our best run, we scored 179 questions correct (35.8%), for which the expected unsorted average precision is 0.358. The maximum possible average precision is 0.726 for this number correct, based on the above formula. Our score of 0.588 represents a ranking ability of .625, indicating a good correlation of confidence and correctness. The top 15 submissions are shown in Table 2, sorted by ranking ability.

## 6 Conclusions and Future Work

We have presented here the first quantitative results from our new PIQUANT question answering system. PIQUANT exploits a multi-strategy and multi-source approach to QA, enabling not only the best approach to be taken on a per-question basis, but the use of mutual reinforcement when multiple agents or sources are used simultaneously. Based on our submissions to TREC and their results, we have shown significant improvements achieved by our approaches over baseline systems. First, we have shown an 14.6% relative gain in average precision score with multiple corpora over a single one, and a further 9.7% relative gain by adding a statistical answering agent. Second, we have identified an effective method for assigning NIL answers to questions based on the confidence values generated by our system. This method identified 63% of all no-answer questions in the test set with minimal false negatives. Third, we have shown that a multi-agent approach to question answering allows us to achieve a good correlation of confidence values and correctness. Our average precision of 0.588 on 179 correct questions achieved 62.5% of the gain achievable by sorting, a significant improvement over the baseline of random sorting.

We have only just begun to incorporate a knowledge

| Submission | AP | % Correct | Ranking Ability |
|---|---|---|---|
| limsiQalir2 | .497 | 26.6 | .657 |
| IBMPQSQACYC | .588 | 35.8 | .627 |
| BBN2002C | .499 | 28.4 | .603 |
| nuslamp2002 | .396 | 21.0 | .569 |
| IRST02D1 | .589 | 38.4 | .559 |
| isi02 | .498 | 29.8 | .555 |
| FDUT11QA1 | .434 | 24.8 | .539 |
| ibmsqa02c | .455 | 29.0 | .461 |
| exactanswer | .691 | 54.2 | .449 |
| ilv02wt | .450 | 30.8 | .392 |
| uwmtB3 | .512 | 36.8 | .392 |
| ali2002b | .496 | 36.2 | .365 |
| aranea02a | .433 | 30.4 | .357 |
| LCCmain2002 | .856 | 83.0 | .168 |
| pris2002 | .610 | 58.0 | .095 |

Table 2: Ranking Ability of Top 15 Submissions

base and inference engine (Cyc) to do sanity checking of answer candidates: the number of times this capability was invoked are too few to do other than say that the approach looks promising. In our future work, we plan to expand PIQUANT's ability to recognize cases when sanity checking is appropriate, improve Cyc's coverage of valid answer ranges, as well as adopt a confidence-based approach to selecting answering agents. Improvements since TREC have led to 16 invocations of the sanity checker on the TREC 2002 question set. These invocations led to one additional correct 1st, 2nd, and 3rd place answers each, validated 4 correct 1st place answers while erroneously validating 3 incorrect 1st place answers, and rejected 122 incorrect answers without any erroneous rejections.

## Acknowledgments

We would like to thank Abe Ittycheriah and Salim Roukos for making available their system and results for experimental purposes, Stefano Bertolo for his help with Cyc integration, and Ruchi Kalra for ontology population. This work was supported in part by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program under contract number MDA904-01-C-0988.

## References

Charles Clarke, Gordon Cormack, and Thomas Lynam. 2001a. Exploiting redundancy in question answering. In *Proceedings of the 24th SIGIR Conference*, pages 358–365.

C.L.A. Clarke, C.V. Cormack, T.R. Lynam, C.M. Li, and McLearn G.L. 2001b. Web reinforced question answering. In *Proceedings of the Tenth Text Retrieval Conference*, pages 673–679.

Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2001. The role of lexico-semantic feedback in open-domain textual question-answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 274–281.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. 2000. Question answering in Webclopedia. In *Proceedings of the Ninth Text REtrieval Conference*, pages 655–664.

Abraham Ittycheriah and Salim Roukos. 2002. IBM's statistical question answering system – TREC-11. In *Proceedings of the Eleventh Text Retrieval Conference*.

Abraham Ittycheriah, Martin Franz, and Salim Roukos. 2001. IBM's statistical question answering system – TREC10. In *Proceedings of the Tenth Text Retrieval Conference*, pages 258–264.

Douglas B. Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11).

George Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11).

Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum, and Vasile Rus. 2000. The structure and performance of an open-domain question answering system. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 563–570.

Marius Pasca and Sanda Harabagiu. 2001. High performance question answering. In *Proceedings of the 24th SIGIR Conference on Research and Development in Information Retrieval*, pages 366–374.

John Prager, Eric Brown, Anni Coden, and Dragomir Radev. 2000. Question-answering by predictive annotation. In *Proceedings of the 23rd SIGIR Conference*, pages 184–191.

John Prager, Dragomir Radev, and Krzysztof Czuba. 2001. Answering what-is questions by virtual annotation. In *Proceedings of Human Language Technologies Conference*, pages 26–30.

John Prager, Jennifer Chu-Carroll, Eric Brown, and Krzysztof Czuba. 2003. Question answering using predictive annotation. In *Advances in Question Answering*. To appear.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 41–47.

Ellen M. Voorhees. 2001. Overview of the TREC 2001 question answering track. In *Proceedings of the 10th Text Retrieval Conference*, pages 42–51.