IBM's PIQUANT II in TREC2004

Jennifer Chu-Carroll, Krzysztof Czuba, John Prager, Abraham Ittycheriah
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
{jencc,kczuba,jprager,abei}@us.ibm.com

Sasha Blair-Goldensohn[1]
Dept. of Computer Science
Columbia University
New York, NY
sashabg@cs.columbia.edu

Introduction

PIQUANT II, the system we used for TREC 2004, is a completely reengineered system whose core functionalities for answering factoid and list questions remain largely unchanged from previous years [Chu-Carroll et al, 2003, Prager et al, 2004]. We continue to address these questions using our multi-strategy and multi-source approach. For "other" questions, we experimented with two alternative approaches, one that uses statistical collocation information for extracting prominent passages related to the target, and the other which is a slight variation of the QA-by-Dossier approach we employed last year [Prager et al, 2004] that asks a set of sub-questions of interest about the target and returns a set of relevant passages that answer these sub-questions. In addition, to address this year's new question format, we developed a question pre-processing component to interpret individual questions against the given target to generate a self-contained natural language question as expected by subsequent components of our QA system. NIST assessed scores showed substantial improvement of our new PIQUANT II system over earlier versions of our QA system both in terms of absolute scores as well as relative improvement compared to the best and median scores in each of the three component subtasks.

The PIQUANT II System

In an effort to create an efficient development and test platform for question answering, we devoted a significant amount of time to system re-engineering. The primary goals of this re-engineering effort were to address the portability, component reusability, and speed of our original PIQUANT system. The system that we used for the TREC-2004 evaluation was built within our new PIQUANT II architecture framework [Czuba, forthcoming]. The main characteristics of the framework are:

1. Full Java implementation.
2. Plug-and-play architecture.
3. Well-defined, standardized APIs for typical QA system components.
4. Distributed client-server deployment.

Our new architecture continues to support our multi-agent approach to QA where different strategies are employed to address different question types. Figure 1 shows a diagram of the PIQUANT II framework as instantiated in the factoid and list subtasks of our TREC 2004 runs. The box labeled "Answer Agents" in the middle of the diagram is the focus of our framework, which allows for plug-and-play of multiple answering agents that make up the core of a QA system. The PIQUANT II framework provides the machinery to execute these answering agents in parallel, and to accumulate and send the results from individual agents to the answer resolution

[1] Part of this work was conducted while Sasha Blair-Goldensohn was a summer intern at the IBM T. J. Watson Research Center.

component. The answer resolution component combines top candidate answers from each agent and produces the top *n* answers that represent the answers of the QA system as a whole. In our TREC 2004 runs, the answer resolution component adopts an equal a priori probability weighting scheme and thus simply sums the agent confidence scores of all semantically equivalent answers and outputs the answer with the highest combined score.

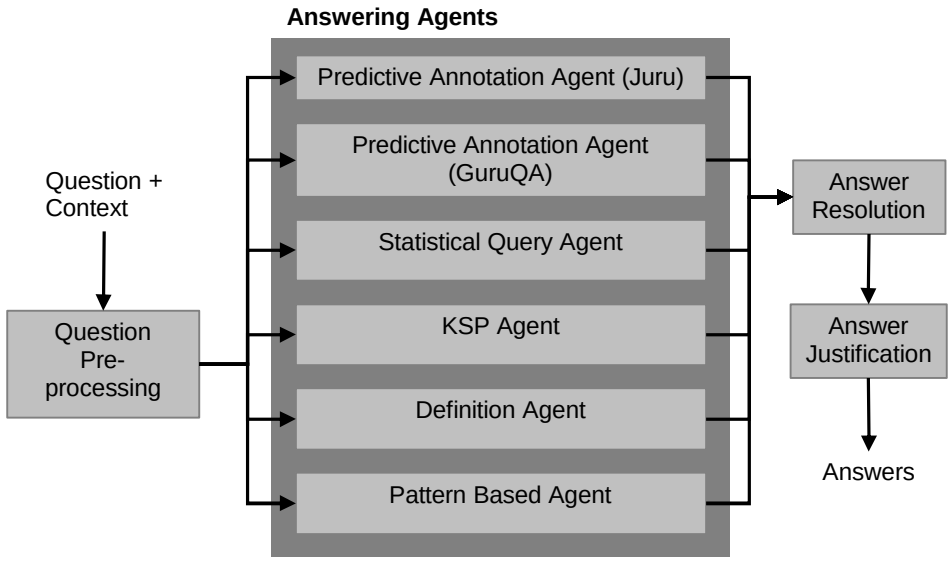

Figure 1 PIQUANT II Framework as Instantiated for TREC 2004 Factoid & List Questions

The answer justification component attempts to locate a passage in a reference corpus that justifies a given answer to a question. This component is useful when an answer to a question was found via other means (such as through a database lookup or on the web), but a "trusted source" is needed to justify the answer. In the case of PIQUANT II in TREC, it is used when an answer is found by our KSP agent, which performs a database lookup of the attribute value of an object for questions such as "*What is the capital of Canada?,*" and a document in the reference corpus is required to support the answer.

The question preprocessor performs any transformation needed on the system input to ensure that answering agents receive their expected input (usually self-contained natural language questions). In the next section, we describe the question preprocessor we developed to address this year's QA task, as well as the new answering agents we have developed in this framework.

PIQUANT II Components in TREC 2004

Question Preprocessor

Since this year's track introduced the notion of a question target and a set of questions related to it, a question preprocessor was required to perform anaphora resolution on each question against the target to produce a self-contained natural language question for subsequent processing. Since we did not have a full anaphora resolution module available, we developed a set of heuristics to handle the different kinds of anaphora we thought were likely to occur. However, prior to performing anaphora resolution, we analyze the target to extract appositional constructs.

The target analysis task focused primarily on extracting proper names from appositional constructs. For example, the target *the band Nirvana* was preprocessed to a modifier *the band* and *Nirvana* that became our new target. The new target was used in anaphora resolution as described below, whereas the modifier was kept and added to the search query for disambiguation purposes.

Our answer selection ranking algorithm makes use of syntactic relationships extracted from the question text; it is thus highly desirable that the question text contains the actual target instead of an anaphor. In order to satisfy this requirement we addressed two anaphora types in questions:

1. Pronominal anaphora including possessives.
2. Definite NPs.

We divided anaphoric pronouns into two groups, those that are unambiguous (*him, his, hers, it, its, their, theirs*), and those that are ambiguous (*her*: possessive vs. accusative). For unambiguous pronouns, we replaced them in the question text with an appropriate form of the target. E.g. for the target *Nirvana*, the question *What is its biggest hit?* is transformed to *What is Nirvana's biggest hit?* For the ambiguous *her*, the question is first parsed using IBM's ESG parser [McCord, 1989] to determine whether it is an accusative or possessive pronoun and the question is transformed appropriately. We assume that only one pronoun is likely to occur in a question, and thus we replace the pronoun earliest in the sentence. We also assume that the pronoun refers to the question target as opposed to an entity in a preceding question.

If the question does not contain a pronoun, it is checked for the presence of definite noun phrases. We assume that definite noun phrases (which we in turn assume begin with a definite article) are all referential if the question target does not appear in the question. In order to enable the right set of relationships to be build for such NPs, we add an *of-PP* to the question. E.g., for the target *IBM*, the question *Where are the company's headquarters?* is transformed to *Where are the company of IBM's headquarters?* Although the resulting English is somewhat odd, our parser handles the transformed questions correctly and an appropriate set of syntactic relationships is generated.

Answering Agents

For the most part, the answering agents we employed this year are algorithmically the same as the agents we used last year, re-engineered to conform to our new architecture. The two exceptions are the Juru-based predictive annotation agent and the profile agent, whose development illustrates some of the key features of the PIQUANT II architecture. The modular architecture and standardized APIs in the PIQUANT II framework enabled component reuse and thus allowed us to rapidly deploy new agents. This was crucial for the development of the profile agent given the short time we had to address the "other" question type before the submission deadline. By reusing existing components as building blocks, the first version of the profile agent was literally built within hours. We then spent the next two weeks experimenting with various concept and passage extraction strategies to obtain the version of the agent used in our submission.

Juru-Based Predictive Annotation Agent

The Juru-based predictive annotation agent implements our previously reported Predictive Annotation strategy [Prager et al., 2000] in our new framework. In doing so, we adopted components that conform to IBM's Unstructured Information Management Architecture [Ferrucci and Lally, 2004], including various named entity annotators, parser, and search engine. The resulting answering agent ended up being significantly different in terms of component strategies and performance from the earlier implementation that was deployed in previous years' TREC evaluations. The primary difference is the use of the Juru search engine developed at IBM's Haifa

Labs, which placed first in the "precision at 10" category in the TREC web track in 2001 [Carmel et al., 2002]. In particular, we used JuruXML [Carmel et al., 2003], an extension of Juru, which supports queries over XML fragments, and enables us to implement our predictive annotation scheme. Juru/JuruXML has different characteristics than the GuruQA passage retrieval engine we previously used in several regards. First, it adopts a tf*idf based ranking scheme, rather than the weighted Boolean query scheme in GuruQA. As a result, the documents/passages retrieved for the same query could differ dramatically. Second, our previous query building process relied quite heavily on morphological and synonym expansions of keywords. However, Juru query syntax does not support treating a group of words/phrases as synonyms in its scoring process. For these reasons and in a general effort to improve the quality of our information retrieval, we developed a new query building component.

Our approach in creating this new query builder was to make a parameterized component which would allow us to easily explore the effects of different query building strategies. These parameters controlled how elements of our question analysis output such as predicted answer type, question keywords/phrases or question semantic type would be included, excluded or required in the query. Using data from previous TRECs, we analyzed the effect of these settings in terms of document/passage recall (i.e. the proportion of documents/passages known to contain correct answers which were retrieved) and end-to-end MRR performance. Although work to optimize these parameters is ongoing, we observed that certain settings improved results on training data, and used those settings for our submitted run. Settings which improved results included:

- Requiring a predictive-annotation token of the predicted answer type
- Requiring the highest-idf single word from the question, based on the idea that this term often serves as an anchor or "selector" term (cf. [Ramakrishnan et al., 2004])
- Including all question keywords (i.e. not using stoplisting but rather allowing Juru's tf*idf ranking to determine term significance)
- Excluding WordNet-based synonym or hypernym expansions (various restrictions and strategies were applied, none improved results)

After Juru returns the top-ranked documents from the query, we apply density-based passage retrieval to identify the most promising passages, and lastly perform answer selection[2].

In terms of accuracy, the new predictive annotation agent is comparable to our previous implementation. However, because of the difference in search engine behavior, there is substantial non-overlap in the actual questions on which each agent scored, we therefore included both implementations of the agent in the QA system we used in our submission.

In terms of run time, the new Java implementation on Linux (with a couple of Windows-dependent components running as servers) averages 7 seconds per question, compared with 2 minutes and 26 seconds per question on our old implementation in Perl/C/Java on AIX.[3] This significant improvement in system speed, although not a factor in the TREC QA track evaluation, is crucial to end-user experience, to enabling more rapid regression tests, and to allowing for more complex processing mechanisms to be adopted in the system.

---

[2] Our question analysis and answer selection components are algorithmically the same as their counterparts in the previous implementation.

[3] Although the predictive annotation agent is a general purpose agent, it does not attempt to answer all question types. To be more comprehensive in coverage, a QA system will include more specialized agents in addition to this agent, which may increase response time (however, as the agents are now executed in parallel, the overall execution time is the maximum time needed by all agents plus some slight overhead).

Profile Agent

The profile agent was the first new agent developed within the PIQUANT II framework, and was designed to select information of interest about a given target from a reference corpus. In contrast to the approach taken by many other systems to date where the system attempts to identify both the definition of the target (expressed, for instance, in a copula construction or as an apposition) and other prominent information of interest (cf. [Blair-Goldensohn et al., 2004; Xu et al., 2004; Echihabi et al., 2004]), our profile agent adopts a statistical collocation-based method and focuses on identifying passages that convey prominent information associated with the target that is difficult to discover based on syntactic information alone.

The profile agent extracts relevant information in a three-stage process. In the first stage, short passages about the target are extracted from the reference corpus. In the second stage, entities that are strongly associated with the target are identified from within these passages. In the third stage, a subset of the extracted passages is chosen to convey the relationship between the selected entities and the target. We experimented with various strategies in all three stages, and in this paper, we briefly outline the strategies used in our submitted run. Note that since we only started to implement this agent two weeks prior to the QA evaluation period, we experimented with different strategies to the best of our abilities at that time. We are aware that there is significant room for future experimentation and improvement and intend to further develop this agent.

The passage extraction process reuses the search component in our Juru-based predictive annotation agent. A search query is formulated based on the given target to retrieve the 100 most relevant documents, from which up to 500 one-sentence passages relevant to the target are selected. In the entity selection phase, the extracted passages are processed by the ESG parser and all common nouns are identified and normalized. The occurrence count for each normalized noun is compared against the expected count based on its idf value in the corpus, and a score is assigned for each noun which is the difference between these two counts. The system then selects as candidate concepts to be included in the output those nouns whose score exceeds a certain predetermined threshold. In the final stage, the profile agent selects as its output a subset of the extracted passages that best conveys the candidate concepts. These passages are selected by considering each candidate concept in ranked order: of all passages that include the currently highest ranked candidate concept, the passage that contains the maximum number of other candidate concepts is selected. All concepts mentioned in the selected passage are removed from the candidate list and the process iterates until either the candidate concept list is exhausted or the maximum number of passages is reached (in our run, the maximum passage count is 20).

QA-by-Dossier Agent

Except in one respect, our use of QA-by-Dossier (QbD) was identical to last year's. QbD consists of building ahead of time a set of auxiliary questions for each broad target type. For example, for PERSON there was "When was X born?", for ORGANIZATION "Who is the CEO of X?", for THING "What does X do?". QbD then calls our QA system recursively so that the factoid-oriented agents can answer these questions. We had about a dozen auxiliary questions per type. Acknowledging that some important nuggets might be missed by this agent (but possibly found by others), we used this approach because analysis of biographies and obituaries had determined that a large percentage of the de facto important definitional information was in the form of answers to such boilerplate questions.

The major difference between our QbD run this year and last (other than the different platform, and that we had no time to learn new thresholds) was that we returned entire sentences instead of exact answers. Our factoid QA-system returns exact answers, and last year for definitional

questions we returned the exact answers prefixed by a term indicating the relationship in the question, for example "born: 1900". Our definitional question results last year were disappointing, and we wondered whether the short nature of our returned answers hurt us because the context was missing, so this year we returned the entire sentence that contained the exact answer. As will be shown in the Analysis section below, we did better with long answers but for an entirely different reason.

System Performance and Analysis

We submitted two runs to the TREC 2004 QA track whose only difference is in the strategy adopted for answering "other" questions. For both runs, the factoid questions are answered by submitting each question to all available factoid agents and weighing the answers proposed by each agent in proportion to their respective confidence scores. We did not adopt a NIL strategy and therefore returned our best answer for each question. For list questions, we observed from last year's results that our system has relatively high precision in its top answers, but does not fare well in recall even if more answers are returned; therefore, we chose to target precision by returning our top 5 answers for each list question. For the "other" questions, run *IBM1* adopted the profile agent for selecting significant passages related to the target, while run *IBM2* used the QA-by-Dossier agent to return passages that answer each sub-question about the target. The results for these runs as scored by the NIST assessors are shown in Table 1.

Run	Factoid	List	Other	Overall
IBM1	.313	.200	.285	.278
IBM2	.313	.200	.227	.263

Table 1 Assessed Scores for Submitted Runs

Performance Analysis

Since our entry this year was based on a re-engineered system with several new strategies employed, we are particularly interested in how our system performed relative to our old system which had been in continuous development over 5 years. Table 2 summarizes the results of comparing our higher-scoring run this year against our performance from TREC 2003. In an attempt to account for possible differences in task difficulty, we further contrast our system performance changes between 2003 and 2004 against changes in the best and median scores of all submitted runs for each of the three subtasks.

	Best Score	Median Score	PIQUANT Score
Factoid			
2003	0.700	0.177	0.298
2004	0.770	0.170	0.313
% change	+10%	-4%	+5%
List			
2003	0.396	0.069	0.077
2004	0.622	0.094	0.200
% change	+57%	+36%	+160%
Other			
2003	0.555	0.192	0.177
2004	0.460	0.184	0.285
% change	-17%	-4%	+61%

Table 2 Comparison of PIQUANT Results against Best and Median Scores of All Runs

Our analysis shows that in absolute numbers, our new system performs better than the old system on all three subtasks, and that in terms of relative improvement, the percent increase figures range widely from 5% to 160%. A closer examination of the performance in each subtask shows that for factoid questions, for which the main difference from last year is the addition of a new implementation of our predictive annotation agent employing a different search strategy, our percent change falls in between the percent changes of the best and median scores, suggesting that our improvement is on target based on the global trend. For list questions, we inherited the improvement in factoid question answering, as well as changed our thresholding strategy to target high precision in our returned answer set. These two factors combined turned out to have a dramatic impact, achieving significantly better performance improvement over the median scoring system (our system's list score last year was only slightly above median). Finally, for the "other" category, our system, through a combination of adopting different selection strategies and returning passage-length answers, achieved a 61% relative improvement over last year, while the best and median scores both decreased.

QA-by-Dossier Performance

We sought to determine whether our long answers to the "other" questions performed better than our exact answers to definition questions last year – in particular we were concerned that the brevity of our exact-answer format might have hurt us in the judging. Last year we achieved 32 vital nuggets out of a total of 207, giving a recall of 0.155. To compute a comparable figure for this year, we needed to simulate last year's operational setup and scoring. To do that, we had to add to the vital nuggets those factoid questions that were paraphrases of those in the QbD auxiliary question sets (since if the same answer is provided in the "other" section as in the factoid, the one in the factoid section gets the score). By our count, we got 46 factoid questions that were in the QbD auxiliary set, plus 12 vital nuggets that were answers to QbD questions. There were a total of 146 QbD factoid questions, and 234 acceptable vital nuggets, giving a recall of (46+12)/(146+234) = 0.153. This is eerily close to last year's figure, but due to the number of variable factors involved, the safest conclusion is simply that there was no perceptible gain in the assessor's ability to detect correct nuggets from sentence-length answers.

However, we did perform better with longer answers than last year, but for a different reason. Our auxiliary questions scored only12 vital nuggets *that were direct answers to these questions*, but 64 vital nuggets that were in the answer sentences "by chance". Likewise, we scored 18 "okay" nuggets as direct answers to the QbD questions, and 76 "by chance". This reinforces a phenomenon that we've been observing in our work with QA, which is that exact-answer QA retrieval systems can be used effectively for passage-retrieval applications.

Summary

In this paper, we described our PIQUANT II system as configured in the TREC 2004 QA runs. Our effort this year focused on a the reengineering of our QA system in order to 1) have a well-designed QA architecture for future development and 2) significantly reduce system response time. For our submitted runs, one of the primary improvements we made was the development of the profile agent, which achieved a 61% relative improvement over our score in the corresponding subtask last year when the trend in all submitted runs shows a decrease. Our change in strategy for the list questions to target high precision in the answer set also appeared to have a significant impact, while our improvement in the factoid task appears to be on target compared to best and median score changes.

Acknowledgments

We would like to thank Dave Ferrucci for helpful discussions, and Elena Filatova for her help in testing the question pre-processing component.

References

S. Blair-Goldensohn, K. McKeown, and A. Schlaikjer, 2004. Answering Definitional Questions: A Hybrid Approach. *New Directions in Question Answering*, M. Maybury (ed).

D. Carmel, E. Amitay, M. Hersovici, Y. Maarek, Y. Petruschka, and A. Soffer, 2002. Juru at TREC 10 -- Experiments with Index Pruning. *Proceedings of TREC2001*, pages 228-236.

D. Carmel, Y. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer, 2003. Searching XML documents via XML fragments. *Proceedings of SIGIR 2003*, pp. 151-158.

J. Chu-Carroll, J. Prager, C. Welty, K. Czuba, and D. Ferrucci, 2003. A Multi-Strategy and Multi-Source Approach to Question Answering. *Proceedings of TREC2002*. Pages 281-288.

K. Czuba. Extendable Plug-and-Play Architecture Framework for Question Answering. Forthcoming.

A. Echihabi, U. Hermjakob, E. Hovy, D. Marcu, E. Melz, and D. Ravichandran, 2004. Multiple-Engine Question Answering in TextMap. *Proceedings of TREC2003.*

D. Ferrucci and A. Lally, 2004. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Journal of Natural Language Engineering*, 10(3-4), pp. 327-348.

M. McCord, 1989. Slot grammar: A system for simpler construction of practical natural language grammars. *Natural Language and Logic*, pp.118--145.

J.M. Prager, E.W. Brown, A. Coden, and D. Radev, 2000. Question-Answering by Predictive Annotation. *Proceedings of SIGIR 2000*, pp. 184-191, Athens, Greece.

J. Prager, J. Chu-Carroll, K. Czuba, C. Welty, A. Ittycheriah, and R. Mahindru, 2004. IBM's PIQUANT in TREC2003. *Proceedings of TREC2003.*

G. Ramakrishnan, S. Chakrabarti, D. Paranjpe, P. Bhattacharyya, 2004. Is Question Answering a Required Skill? *Proceedings of WWW2004*, pp. 111-120.

J. Xu, A. Licuanan, and R. Weischedel, 2004. TREC2003 QA at BBN: Answering Definitional Questions. *Proceedings of TREC2003.*