IBM's PIQUANT II in TREC2005

Jennifer Chu-Carroll† Krzysztof Czuba* Pablo Duboue† John Prager†

† IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
{jencc, duboue, jprager}@us.ibm.com
* Google Inc.,1440 Broadway, 21st Floor, New York, NY 10018, USA
kczuba@google.com

Introduction

This year, the PIQUANT II system we used in the TREC QA track is an improved version over the reengineered system we used in last year's entry [Chu-Carroll et al., 2005]. Our system adopts a multi-agent approach to question answering. In this framework, a question is submitted to multiple agents, each adopting a different question answering strategy and/or consults a different information source to produce a set of answers, which are then combined using a voting scheme to determine the overall system's answer(s) to the question. In our 2005 system, we have made improvements along several dimensions, by improving the performance of select answering agents, by developing two new agents, and finally, by improving our answer resolution algorithm for combining answers from individual agents. In this paper, we describe these improvements and their impact on the factoid, list, and other subtasks in the main task.

PIQUANT II System Overview

As described in [Chu-Carroll et al., 2005], PIQUANT II is built on a modular and extensible architecture that supports component plug-and-play, distributed client-server deployment, and supports well-defined APIs for typical QA system components. Figure 1 shows how PIQUANT II was configured for the TREC 2005 factoid and list subtasks.

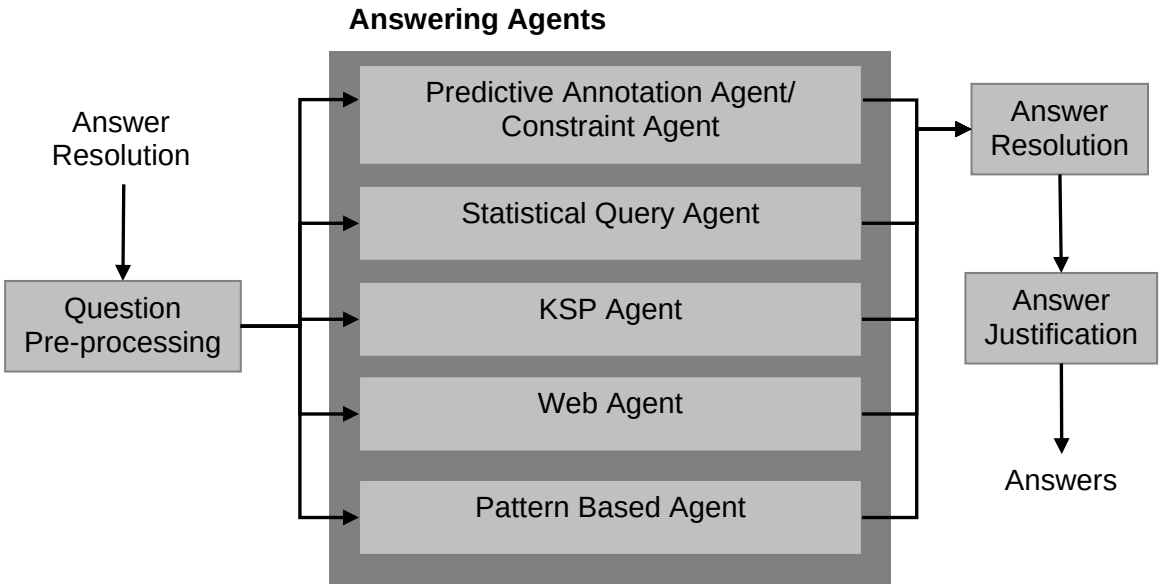

Figure 1 PIQUANT II as Configured for TREC 2005 Factoid & List Questions

The question pre-processing component takes a question, which may contain referring expressions or ellipses, and a target as input, and performs any transformation needed to produce a self-contained question. This question is then sent to five answering agents in parallel. Of the five answering agents employed by our TREC 2005 system, the statistical query agent and the KSP agent, which answers questions from structured knowledge sources, remain unchanged from previous year's system. We improved the performance of the predictive annotation agent and increased the coverage of the pattern-based agent. Furthermore, we developed a web agent that utilizes the web as an information source and a constraint agent based on, and in run IBM05C3PD used instead of, the predictive annotation agent and employs an automatic answer verification process to improve accuracy. The answers returned by each of the five agents are combined using a machine-learned voting scheme in our answer resolution module. Finally, the answer justification component attempts to locate a passage in a reference corpus to justify an answer found from an alternative source (such as in our structured database for the KSP agent or on the web for the web agent).

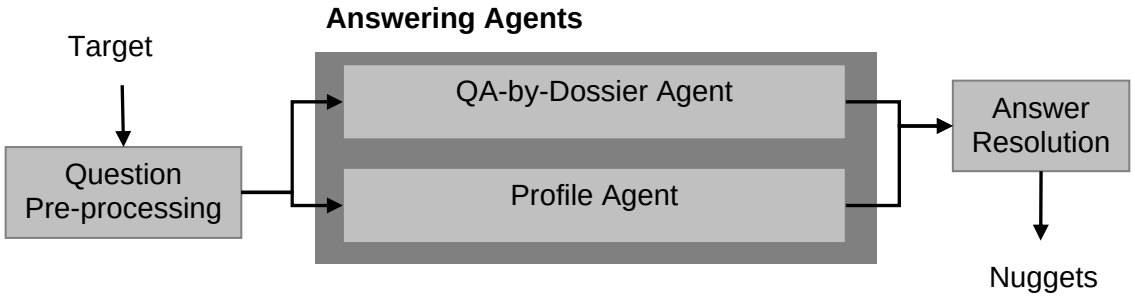

Figure 2 PIQUANT II as Configured for TREC 2005 Other Questions

Figure 2 shows PIQUANT II as it was configured for answering questions in the “other” subtask. In addition to improving the two agents we developed for this subtask in 2004, which were each used in one of the two runs we submitted last year, we developed an answer resolution component to combine the nuggets returned by both agents to produce a combined nugget list.

PIQUANT II Components for Factoid/List Questions

Question Preprocessor

The goal of the question preprocessor in the factoid and list question setting is to resolve referring expressions and ellipses against the question target to produce a self-contained natural language question for subsequent processing. In general, we eliminate anaphors and introduce the target so that a better search engine query could be formulated. Our assumption was that anaphors in a question always refer to the target and not to an answer to a previous question in a sequence. This has been borne out in the question set in which only a handful of anaphors could be interpreted as referring to an answer.

Anaphors were replaced with the target string in a way that attempted to keep the resulting question grammatical. In the case of the ambiguous pronoun "her", we relied on a deep syntactic parser, ESG [McCord, 1989], to identify it as a possessive or accusative. We replaced an occurrence of a possessive pronoun with "X's" where X is the target. All other pronominal anaphors were replaced with the target.

Based on the sample event targets that were available as training data, we implemented a restriction on the maximum length of targets that could replace anaphors. Another restriction was on the capitalization within a target to prevent targets like "The return of Hong Kong to China" from getting embedded in questions.

Answering Agents

In this section, we discuss enhancements we made to the predictive annotation agent as well as describe a newly developed agent, the constraint agent.

Predictive Annotation Agent

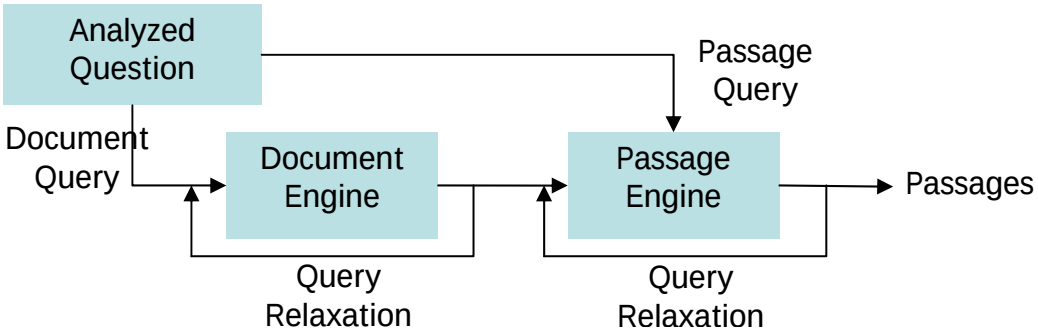

Figure 3 Retrieval Component with Feedback Loops

For the most part, the question analysis and answer selection components of our predictive annotation agent remain unchanged from our 2004 system. The key improvements we made were in the document/passage retrieval stage in the following two aspects. First, our earlier system treats the document retrieval and passage ranking processes as a one-stage process. A single query is given to this retrieval component and this query is both used to retrieve relevant documents using the JuruXML search engine [Carmel et al., 2003] and to identify the most relevant 1-3 sentence passages from these documents. In our new retrieval architecture, which is shown in Figure 3, we separated the document retrieval stage from the passage ranking stage and allowed for different queries to be considered for each of the two processes so that certain query terms which may be present to set the context of the document, but which may not be repeated in all 1-3 sentence windows, can be taken into account in the document retrieval stage but not in passage ranking. Second, in our new architecture, we introduced feedback loops [Pasca and Harabagiu, 2001] to allow the system to initially target high precision queries and gradually relax the queries should they prove to be overly constrained.

Constraint Agent

The constraint agent is an extension of the work first presented in [Prager et al, 2004] and is the first time we employed it in any of our TREC runs. We used the constraint agent, which invokes the predictive annotation agent recursively only on factoid questions, although in principle it could be used on any type. Simply put, it operates by generating a candidate answer set in the usual way, then inverts the question and substitutes each candidate answer in turn, and examining if the inverted answer matches a term in the original question. This can be illustrated by means of an example.

Question 110.2 was (substituting the target) “When was the club Lions Club International founded”? The internally generated initial candidate answer set was, in order of preference, “1997, 1917, …”. The constraint agent automatically generated the inverted question “What club was founded in X?”, and substituted in turn the values in the initial candidate answer list for X. These new inverted questions are then submitted again to the predictive annotation agent and the top answer for “What club was founded in 1917?”

was indeed the Lions Club International, thus providing confirming evidence for 1917. On the other hand, the answer for "What club was founded in 1997?" was another entity, thus refuting the answer 1997. The recomputed score for 1917 was higher than the recomputed score for 1997, so the former was returned.

The algorithm used for determining whether and how to adjust the original candidate list consists applying a decision tree to the confidences of the answers in the forward and inverted directions. The parameters and thresholds used in the decision tree were determined by machine learning over training data. The particular implementation of the constraint agent we used for TREC just looked at the top two candidate answers.

Answer Resolution Component

In our TREC 2004 system, we adopted an answer resolution strategy that combines answers returned by each answering agent using an equal a priori probability. In 2005, we developed a new answer resolution component that uses a linear combination of the agent's a priori weight based on its prior performance and the confidence score the agent assigned to an answer. In addition, through experimentation with parameter training based on questions with different answer types, we found that for our set of five agents and their respective performance, adopting a different set of a priori weights for questions seeking dates as answers is beneficial.

PIQUANT II Components for "Other" Questions

QA-by-Dossier Agent

The QA-by-Dossier agent is built on the observation that some of the interesting properties of a target entity can be predicted by simply knowing the type of the entity. For example, knowing the target is a person, then occupation, birth and death dates and other life-cycle information are probably going to be at the very least "okay", if not "vital". Thus by preparing a number of *subquestions* (such as "When was X born?", "Where is Y headquartered?") keyed on type we could call our QA system recursively and generate information nuggets to return. We understood this strategy would not necessarily capture surprising or unexpected information nuggets, but we hoped that our profile agent, using a collocation plus idf-based approach, would cover those cases.

In 2004 the QA-byDossier agent did not work terribly well, primarily because the information on the assessors nugget lists was very heterogeneous, and did not seem to follow any consistent usage model (such as preparing an obituary or encyclopedia article, on which our questions were based). For 2005 we hoped that after discussions of these issues both by us and others [Chu-Carroll et al., 2004, Hildebrandt et al., 2004], the nugget lists would be more consistent, so we continued this approach, but tried to expand the nugget net by using a broader and deeper set of questions.

In TREC 2004 we simply classified the target as Person, Organization or Thing, with a small number (3-12) of corresponding questions. For 2005 we classified the target into one of 20 entity categories, including subtypes of the former three (e.g. Inventor, Athlete, Entertainer, Writer) and orthogonal ones (e.g. Place, Religion). Questions were

generated by hand for each of the categories. These were arranged in a hierarchy, so that Inventor had Inventor-specific questions ("What did X invent?"), but also inherited from Person ("Where was X born?"). In an attempt to capture the surprising facts, we asked questions like "What crime did X commit?", again keyed on type (Person in this case), relying on the fact that if that person was not a criminal, the QA system will return either no answer or an answer with very low confidence (and thus will not be chosen for output by the QA-by-Dossier agent).

Profile Agent

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.

In our 2004 system, in the entity selection phase, the extracted passages are processed by the ESG parser and all common nouns are identified and normalized. Those nouns that occurred more frequently than expected based on their idf value are selected as candidate entities. In our 2005 system, we improved upon this entity selection process by considering not just co-occurrence information of nouns, but all concepts in the passages and weighing those concepts that are syntactically related to the target more strongly than others.

System Performance and Analysis

We submitted three runs to the TREC 2005 QA track, whose results as scored by the NIST assessors are shown in Table 1. The first run, IBM05C3PD employed five answering agents for factoid and list questions as shown in Figure 1, with the constraint agent selected to represent the constraint/predictive annotation agent alternatives. When combining agent answers, the answer resolution component assigned different a priori weights to agents based on answer types, and in answering the "other" questions, both the QA-by-Dossier agent and the profile agent were employed. In runs IBM05L1P and IBM05L3P, the predictive annotation agent was used instead of the constraint agent in both runs, and only the profile agent was used for the "other" questions.

Run	Factoid	List	Other	Overall
IBM05C3PD	.323	.131	.206	.246
IBM05L1P	.323	.131	.192	.242
IBM05L3P	.326	.131	.192	.244

Table 1 Assessed Scores for Submitted Runs

As our results show, the impact of the different system configurations on our overall score is minimal. The largest difference is in the addition of the QA-by-Dossier agent in our first run, affecting 5 out of 65 questions[1] and increasing our F-score from 0.192 to 0.206. Note that some of the output of the agent was redundant with that of the profile

[1] We have discovered after TREC that a bug in the expected input for the QA-by-Dossier agent resulted in decreased impact of the agent on our TREC results.

agent and some were precluded in scoring because the same question was asked, albeit with different words, in the factoid section. The QA-by-Dossier agent performance also suffered because we had no subquestions for the Event type target, which turned out to be quite prevalent.

Across the factoid questions, the constraint agent materially changed only 12 answers, but of these only two went from "Wrong" to "Right" or "Inexact", and two went the other way. Follow-up experimentation has shown that the constraint agent works better when the scores associated with candidate answers are close to confidences or probabilities of being correct. However, in our system development prior to TREC the only consideration put on the scores was that the better answer should have a higher score, rather than represent confidence in the absolute sense, which appears to be necessary for the constraint mechanism to be effective. We are currently addressing this issue to improve the performance of the constraint agent.

Summary

In this paper, we described our PIQUANT II system as configured in the TREC 2005 QA runs. Our efforts this year focused on algorithmic improvements of individual answering agents as well as development of new agents employing different question answering strategies. Our resulting system ranked 3rd out of 30 participating systems in the TREC QA main task.

Acknowledgments

This work was supposed in part by the Disruptive Technology Office (DTO)'s Advanced Question Answering for Intelligence (AQUAINT) program under contract number H98230-04-C-1577.

References

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, K. Czuba, J. Prager, A. Ittycheriah, and S. Blair-Goldensohn, 2005. IBM's PIQUANT II in TREC 2004. *Proceedings of TREC2004.*

W. Hildebrandt, B. Katz, and J. Lin. Answering definition questions with multiple knowledge sources. *Proceedings of HLT/NAACL 2004.*

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

M. Pasca and S. Harabagiu, 2001. High Performance Question/Answering. *Proceedings of SIGIR 2001*, pp. 366-374.

J. Prager, J. Chu-Carroll, and K. Czuba, 2004. Question answering using constraint satisfaction: QA-by-Dossier-with-Constraints. *Proceedings of ACL 2004*, pp. 575-582.