The QUANTUM Question Answering System at TREC-11

Luc Plamondon Guy Lapalme
RALI/DIRO, Université de Montréal
CP 6128, Succ. Centre-Ville
Montréal, Québec, Canada, H3C 3J7
{plamondl, lapalme}@iro.umontreal.ca

Leila Kosseim
Concordia University
1455 de Maisonneuve Blvd. West
Montréal, Québec, Canada, H3G 1M8
kosseim@cs.concordia.ca

Abstract

This year, we participated to the Question Answering task for the second time with the QUANTUM system. We entered 2 runs for the main task (one using the web, the other without) and 1 run for the list task (without the web). We essentially built on last year's experience to enhance the system. The architecture of QUANTUM is mainly the same as last year: it uses patterns that rely on shallow parsing techniques and regular expressions to analyze the question and then select the most appropriate extraction function. This extraction function is then applied to one-paragraph long passages retrieved by Okapi to extract and score candidate answers. Among the novelties we added to QUANTUM this year is a web module that finds exact answers using high-precision reformulation of the question to anticipate the expected context of the answer.

1 Introduction

TREC-11 marks the second year of existence of QUANTUM, the QUestion ANswering Tool of the University of Montreal. As for last year, we used QUANTUM to participate to the main and the list task, and we did not enter the context task. This year's version of QUANTUM is similar in essence to last year's version, but we enhanced specific modules that provided poor performances last year and we added a module to search for exact answers on the web. Following the conclusions we came to last year [1, 2], we decided to drop our own information retrieval system to rely solely on Okapi[1] [3] since the latter led to clearly better results. Also, we fine-tuned our answer extraction functions by introducing weights in the computation of answer scores.

The TREC-10 version of QUANTUM is described in [1] and an analysis of the results is presented in [2]. We summarize some sections that are still relevant to the understanding of this year's version, while we delve into the new features in more detail.

[1]Okapi-Pack: www.soi.city.ac.uk/~andym/OKAPI-PACK

2 Components of Questions and Answers

QUANTUM uses the same technique for question analysis as last year [2]. To see how we decompose a question, let us consider question *#302 – How many people die from snakebite poisoning in the US per year?* and its answer. As shown in Fig. 1, the question is decomposed in three parts: a *question word*, a *focus* and a *discriminant*, and the answer has two parts: a *candidate* and a variant of the question *discriminant*.

The *focus* is the word or noun phrase that influences our mechanisms for the extraction of candidate answers (whereas the *discriminant*, as we shall see in Sect. 4.4, influences only the scoring of candidate answers once they are extracted). The identification of the focus depends on the selected extraction mechanism; thus, we determine the focus with the syntactic patterns we use during question analysis. Intuitively, the focus is what the question is about, but we may not need to identify one in every question if the chosen mechanism for answer extraction does not require

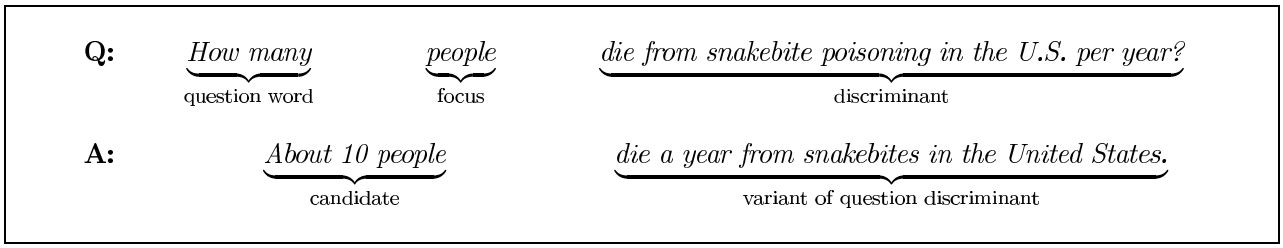

Figure 1: Example of question and answer decomposition. The question is from TREC-9 (# 302) and the answer is from the TREC document collection (document LA082390-0001).

it.

The *discriminant* is the remaining part of a question when we remove the question word and the focus. It contains the information needed to pick the right candidate amongst all. It is less strongly bound to the answer than the focus is: pieces of information that make up the question discriminant could be scattered over the entire paragraph in which the answer appears, or even over the entire document. In simple cases, the information is found as is; in other cases, it must be inferred from the context or from world-knowledge.

We use the term *candidate* to refer to a word or a small group of words, from the document collection, that the system considers as a potential answer to the question. Candidates are usually noun phrases (see Sect. 4.3.1 for a discussion on exact answers).

3 Runs Submitted at TREC-11

This year, we participated to the main task and the list task. We developed two versions of QUANTUM for the main task: one version that does not make use of the web, and one that does. For the list task, only the no-web version was ready by the time of the competition. So only one run was submitted for the list task and two for the main task.

4 Architecture of the Core System

The current version is a continuation of last year's system but with enhancements suggested in our analysis of last year's results and with some changes required by the new specifications of the 11th edition of TREC (exact answer, single answers, etc.). After analyzing last year's performance, we concentrated our efforts in improving the question analysis module to correctly analyze more questions and the scoring module to better score the candidates. In particular, we wanted to better weight the contribution of WordNet and the named-entity tagger depending on the type of question. Last year, our strategy to insert NIL answers in our candidate set dropped our score significantly (from 0.223 to 0.199). We believe that our strategy was sound, but our scoring of the candidates was such that we would have been better off without inserting NIL answers. This is why, this year, we did not attempt to insert NIL answers.

Let us step through the 4 basic steps of the system: question analysis, passage retrieval and tagging, candidate extraction and candidate scoring.

4.1 Question Analysis

To analyze a question, we use a tokenizer, a part-of-speech tagger and a noun-phrase chunker (NP-chunker). QUANTUM then applies a set of handmade patterns based on words, on part-of-speech tags and on noun-phrase tags, in order to select the most appropriate function for answer extraction. Last year, only 40 such patterns were used, but they correctly classified 88 % of the 492 TREC-10 questions. To increase the performance of the classification module, we added 20 more patterns to account for last years mistakes and new question formulations. Once the question is classified, an extraction function determines what criteria a group of words from a document should satisfy to constitute a valid answer candidate; for example, a *location* should begin with a capital letter while a *measure* should include a number and a measure unit. We also added a *synonym* extraction function to our last year's set of 11 functions (Table 1) because some questions in the TREC-10 corpus required it. Extraction functions such as *definition* were less useful this year because the questions were mainly named-entity targeted.

Like last year, each function triggers a search mechanism to identify candidates in a passage based on the passage's syntactic structure or the semantic relations

of its component noun phrases with the question focus. More formally, we have $\mathcal{C} = f(\rho, \varphi)$, where f is the extraction function, ρ is a passage, φ is the question focus and $\mathcal{C}$ is the list of candidates found in ρ. Each element of $\mathcal{C}$ is a tuple (c_i, d_i, s_i), where c_i is the candidate, d_i is the number of the document containing c_i, and s_i is the score assigned by the extraction function.

4.2 Passage Retrieval and Tagging

The extraction of candidates is a time-consuming task. Therefore, we select the shortest, albeit most relevant, passages of the document collection before we begin answer extraction. To do so, we use the Okapi system to retrieve variable-length passages. Passage retrieval has not changed from last year; however, experiments showed that our own fixed-window IR did not achieve as good results as Okapi. So this year, we exclusively used Okapi. Okapi is an information retrieval engine that has the ability to return relevant paragraphs instead of whole documents [3]. We feed it with the question as a query and we set it up so that it returns 30 one-paragraph-long passages (the average length of a passage, or paragraph, is 350 characters).

Since the answers to the TREC-11 factual questions are usually short noun-phrases (Sect. 4.3.1), we run our NP-chunker on the most relevant passages. Our chunker looks for specific sequences of part-of-speech tags, which are given by our tagger. In addition, we run a named entity extractor on the passages because the candidates sought by extraction functions such as *person*(ρ), *time*(ρ) and *location*(ρ), are named entities. For this step, we use the Alembic Workbench system developed at Mitre Corporation for the Message Understanding Conferences (MUC). Amongst all the named entity types that Alembic can recognize [4], we currently use only PERSON, ORGANIZATION, LOCATION, DATE and TIME entities.

The tagged passages are then passed to the previously selected extraction function to identify and score candidate answers.

4.3 Extraction of Candidates

Given the extraction function f chosen after question analysis, the question focus φ and a set of tagged passages ρ_j, candidates c_i are extracted along with their document number d_i and their score s_i (Sect. 4.1). To do so, each function is implemented by a set of search strategies that involve words, part-of-speech tags, semantic relations (mainly hypernym/hyponym relations given by WordNet) and named entities identified by Alembic. Table 1 presented earlier shows some examples of what extraction functions look for. During the extraction phase, we seek a high recall rate, no matter whether candidates are cited in a context that matches the question discriminant; we shall use a combination of scores to account for the context later.

4.3.1 Answer Exactness

A major difference with the TREC-10 QA track is that answers must be exact. This means there is no limit imposed on the length of an answer string, but the string must not contain an incomplete answer nor must it contain more information than requested by the question. What is considered "too much" or "not enough" was not clearly defined at the time the competition was held; the problem was left to the assessors' judgment. Another consideration for having exact answers, although not relevant to TREC, is that potential users of QA systems are more likely to find *the red, green and white flag* more pleasant to read than *), that the red, green and white flag of the Ital.*

We decided to keep the strategy we employed last year, which is to extract all the noun-phrases in a retrieved passage and then to test, using the selected extraction function, whether each of them is an interesting candidate. That means that a candidate is always at least a complete NP and it never encompasses more than one NP, except in a limited number of cases that we explicitly foresaw when writing the extraction functions. We are aware that this is not suitable to all questions but we believe it is a simple way to achieve a satisfying level of answer exactness most of the time.

There are cases where a NP is not long enough (our definition of a single NP does not include conjunctions of embedded NP), typically when (1) a question seeks more than one entity: *#1422 – What two European countries are connected by the St. Gotthard Tunnel under the Alps?* A: *Switzerland and Italy*[2] and (2) the answer is a title or a quote: *#1832 – What did Walter Cronkite say at the end of every show?* A: *"and that's the way it was"*. There are also cases where a NP can be too long. For example, *Louise Veronica Ciccone* would be an inexact answer to question *#1723 – What is Madonna's last name?* A: *Ciccone*.

After examining answer patterns for TREC-8, TREC-9 and TREC-10 questions, we estimated that

[2]Even though this question has a list-task style, it is part of the main task.

Function	**Example of question and sample of answer patterns**
definition(ρ, φ)	**Q:** *#897 – What is an atom?* $(\varphi = atom)$ **A:** <hypernym of *atom*>, <*atom* or hyponym of *atom*> **A:** <*atom* or hyponym of *atom*> (<hypernym of *atom*>) **A:** <*atom* or hyponym of *atom*> `is` <hypernym of *atom*>
specialization(ρ, φ)	**Q:** *#1684 – What card game uses only 48 cards?* $(\varphi = card\ game)$ **A:** <hyponym of *card game*>
cardinality(ρ, φ)	**Q:** *#1761 – How many black keys are on the piano?* $(\varphi = black\ keys)$ **A:** <number> <*black keys* or hyponym of *black key*>
measure(ρ, φ)	**Q:** *#1715 – How much vitamin C should you take in a day?* $(\varphi = vitamin\ C)$ **A:** <number> <hyponym of *unit*> `of` <*vitamin C* or hyponym of *vitamin C*>
attribute(ρ, φ)	**Q:** *#1420 – How high is Mount Kinabalu?* $(\varphi = high)$ **A:** Various patterns
synonym(ρ, φ)	**Q:** *#1651 – What is another name for the North Star?* **A:** *the North Star*, `also known as` <NP> **A:** <NP>, `also known as` *the North Star*
person(ρ)	**Q:** *#1424 – Who won the Oscar for best actor in 1970?* **A:** <PERSON named entity>
time(ρ)	**Q:** *#1676 – When was water found on Mars?* **A:** <TIME named entity> **A:** <hyponym of *time_period*>
location(ρ)	**Q:** *#1483 – Where is the highest point on earth?* **A:** <LOCATION named entity>
manner(ρ)	**Q:** *#1446 – How did Mahatma Gandhi die?* **A:** Not implemented for TREC
reason(ρ)	**Q:** *#902 – Why does the moon turn orange?* **A:** Not implemented for TREC
object(ρ)	Default function **A:** <NP>

Table 1: Extraction functions, examples of TREC questions and samples of answer patterns. Hypernyms and hyponyms are obtained using WordNet, named entities are obtained using Alembic and NP tags are obtained using an NP-chunker. When we mention the focus in an answer pattern, we also imply other close variants or a larger NP headed by the focus.

only 2% of the questions could not be answered by a NP. Therefore, we decided to make no significant change to our system regarding answer exactness.

4.4 Scoring of Candidates

The final score of a candidate is computed as in Eq. 1:

$$score = \alpha \cdot extraction\ score + (1 - \alpha) \cdot passage\ score \quad (1)$$

with α, the *extraction score* and the *passage score* ranging from 0 to 1 (we describe the *extraction score* and the *passage score* below).

We dropped the *proximity score* term that we used last year. This 3rd term was meant to favor candidates that were surrounded by question keywords but it did not have a noticeable influence in the tests we conducted this year.

We found the optimal value $\alpha = 0.75$ by maximizing the performance of QUANTUM on a subset of TREC-8, TREC-9 and TREC-10 questions. We discarded no-answer questions and questions that QUANTUM does not analyze correctly.

The final scores given by Eq. 1 range from 0 to 1. Last year, scores awarded by QUANTUM did not have an upper bound, so candidates extracted using different extraction functions were hardly comparable. The comparability of scores seemed a prerequisite for detecting no-answer questions using a threshold (Sect. 4.5) and for a final re-ordering of the questions based on confidence (Sect. 6).

4.4.1 The Extraction Score

The *extraction score* measures how much a candidate satisfies various criteria that all valid candidates should meet. The set of criteria is specific to each type of question, thus to each extraction function. Criteria are weighted and a candidate does not have to satisfy all of them. Eq. 2 shows the criteria for the *time* function:

$$\text{score}_{time} = \max(entity, hyponym) \cdot penalty \quad (2)$$

with $entity = 0.5$ if the candidate has been tagged as a TIME named entity by Alembic ($entity = 0$ otherwise), $hyponym = 0.25$ if the candidate is a hyponym of the WordNet synset *time_period* or of another selected synset ($hyponym = 0$ otherwise), and $penalty = 0.75$ if the candidate contains one of the question keywords ($penalty = 1$ otherwise so that *score* is not reduced).

The values of the parameters for the 12 extraction functions were found by maximizing the performance of QUANTUM on the same question subset as described above. An outcome of the optimization is the reduction of the influence of WordNet to the benefit of Alembic for the *person* function (WordNet: 0, Alembic: 1), for the *location* function (WordNet: 0, Alembic: 1) and for the *time* function (WordNet: 0.25, Alembic: 0.75). This was expected because our analysis of last year's results led us to the conclusion that WordNet was rather a source of noise when a named entity extractor could be used instead.

4.4.2 The Passage Score

While the extraction score is concerned only with the form and type of a candidate, the *passage score* attempts to take into account the supplemental information brought by the question discriminant. It measures how confident we are in the passage where the candidate is found. For this measure, we use the score given to the passage during its retrieval by Okapi. However, this year, we normalize the score of each passage over the score of the best-scoring passage to have a *passage score* — and thus a final score (Eq. 1) — between 0 and 1. Since the question discriminant is likely to appear in the text under a slightly different form and to be scattered over several sentences around the sought candidate, we believe that an IR engine is the best tool for measuring the concentration of elements from the discriminant in a given passage.

4.5 No-Answer Questions

At TREC-10, we measured the score drop between the ranked candidates for a question and we inserted a NIL answer when the score drop was higher than a threshold. This meant the system would rather say there is no suitable answer in the document collection than propose any candidate at a worse rank. We had chosen to use a threshold on normalized score drops instead of a threshold on absolute scores because different extraction functions would use different score scales. The major drawback of that method is the impossibility to propose a NIL answer at first rank (unless QUANTUM finds no candidate at all, which seldom happens because the extraction functions are very permissive).

This year, we attempted to use scoring methods that produce comparable final scores no matter how the candidates are extracted. Our goal was to set a threshold on the final score below which a NIL answer would be preferred. Unfortunately, because of the small number of criteria used when computing

the extraction score and because of their boolean behaviour, the scores of all the candidates tend to cluster around a few values. We observed that the scores of first-rank candidates ranged from 0.5 to 1, with about one third of them at 0.5. Thus, a NIL insertion based on a score threshold would have resulted in a NIL answer for 33 % of the TREC-10 questions, which seemed too far from reality (10 %) to be an improvement to the system. Therefore, we decided not to detect no-answer questions.

5 Architecture of the Web Module

Following several systems from last year, and because TREC-11 questions are of general domain, the web proved to be an interesting source of answers [5, 6]. We present here how a new module of QUANTUM makes a simple use of the web to retrieve exact answers.

Our use of the web differs from most of last year's participants. To use the web, two strategies could be used: either aiming at high recall or high precision. High recall would mean retrieving a large list of candidate answers and using a good scoring strategy to rank the candidates. This is similar in essence to answer redundancy. We decided to go for the other approach: high precision of the answers at the expense of recall.

We do not seek supplemental documents from the web to complement the set of documents retrieved from the TREC collection, because QUANTUM has enough answer candidates from the TREC collection. The problem is to correctly identify the answer from the list of candidates and score it such that it is positioned at the top of the list. So we use the web to retrieve candidates only through high-precision methods in such a way that the web does not return an answer often, but when it does, the answer is expected to be the correct one. To do so, we look on the web for an exact sentence that could naturally be the formulation of the answer to the question.

To formulate an answer pattern, the TREC question is turned into its declarative form using a set of hand-made patterns. For instance, *#1697 – Where is the Statue of Liberty?* is reformulated as "`the Statue of Liberty is <LOCATION>`". We call this reformulation the *expected context of answer* because we expect to find this pattern, with the correct answer in place of `<LOCATION>`, at least once on the web.

We then use *Yahoo!* to search for web pages that contain the known part of the expected context (here, the phrase "`the Statue of Liberty is`") and we identify answer candidates by unification. We do simple validity checks on candidates, such as testing the length of a candidate or whether a candidate for a location begins with a capital letter, but as for now the tests are not as sophisticated as the extraction functions displayed in Table 1.

The web module can identify about 10 general types of answers (*eg.* `<LOCATION>`, `<NUMBER>`, `<CLAUSE>`, ...). However, by using a conjunction of expected contexts, we can impose more constraints on a candidate. For example, the question *#1851 – Which country colonized Hong Kong?* is reformulated as "`<CLAUSE> colonized Hong Kong`", where `<CLAUSE>` can be any string. To further restrict the candidate, we impose a second context for the candidate to satisfy: "`<CLAUSE> is a country`". The search thus becomes a conjunction of the two strings "`<CLAUSE> colonized Hong Kong`" and "`<CLAUSE> is a country`" (they can be found in separate web pages).

We score the candidates once they are extracted. A candidate that passes the validity checks starts with a score of 0.65. For each additional occurrence of the candidate found in any of the expected contexts derived from the question, the difference to one is divided by 2, thus raising the score to 0.825, then 0.9125 and so on. A candidate that does not meet the validity criteria but appears where an answer was expected receives a score of 0.1. Each additional occurrence of the candidate boosts the score by 0.1, as long as the resulting score does not exceed 0.6. Therefore, a presumably invalid candidate never has a higher score than a presumably valid candidate.

After we find an answer on the web, we use Okapi to perform a search in the TREC-11 document collection in order to have a document number to accompany the candidate. For the questions where either no candidate is found on the web or none of them can be matched with a document number, QUANTUM proceeds without the web module, as described in Sect. 4.

To test the performance of the web module, we conducted experiments with TREC-9 and TREC-10 questions. In the TREC-9 and TREC-10 document collection, we seldom find an expected context (we do for only 10 % of the questions). However, when we search on the web, we find at least one occurrence of an expected context for 43 % of the questions. Of these, the answer identified by unification is correct 51 % of the time, and 45 % of them appear at first rank. In clear, 10 % of the TREC questions are cor-

Run name	# right	Score	Max	Min
UdeMmainNoW	37	0.080	0.266	0.003
UdeMmainWeb	29	0.057	0.222	0.001

Table 2: Confidence-weighted scores of the 2 main-task runs compared to the maximum and minimum possible scores given the number of correct answers.

rectly answered only by searching for expected contexts of answers on the web and by performing basic validity checks.

We also tried to run QUANTUM with and without the web module, and then to keep the best-scoring candidate of either version. Results were worse than with the web module alone, probably because scores obtained with expected contexts and with Eq. 1 should be weighted to be appropriately comparable.

6 Confidence

Answers for the main task have to be submitted to NIST in decreasing order of confidence. To do so, we simply sort them according to their score. Table 2 shows how the sorting affected our runs by comparing their confidence-weighted score to a hypothetical maximum score (all correct answers at the top of the sorted list) and minimum score (all correct answers at the end of the sorted list), given the number of correct answers. Our ordering could be improved considerably.

7 List task

The only supplemental difficulty of the list task is the identification of the desired number of items specified in the question. We had to adapt the new question analysis patterns we added this year in the same way that we adapted last year's main task patterns for the list task. Once the system has analyzed a list question with the appropriate set of patterns, it proceeds exactly as for the main task (unfortunately, only the no-web version described in Sect. 4 was ready for TREC-11), except that it keeps the desired number of candidates instead of only the best one. For simplicity, duplicate candidates are eliminated even if they come from different documents (they might not be real duplicates because one candidate might be supported by its document and the other not).

8 Discussion

When we conducted our pre-competition tests on the TREC-10 question corpus, we estimated that the no-web version of QUANTUM could correctly answer 14 % of the questions, and that the web version would perform even better with a correct answer for 17 % of the questions. This was an improvement over last year's version of QUANTUM, which correctly answered 12 % of the questions (this data comes from last year's best run — with 50-character answers, but keeping only 1 answer per question). However, our estimations were made using an automated evaluation procedure that could not detect unsupported answers nor inexact ones (the *lenient* evaluation).

Table 3 shows the official evaluation details of our 2 main-task runs. We will exclude the effect of confidence weighting from our analysis and we will use Table 4 to compare a strict evaluation (the percentage of right answers) against a lenient evaluation that includes inexact and unsupported answers (as an automatic evaluation script would do).

The lenient evaluation in Table 4 is closer to our pre-competition estimations. The web module is clearly an improvement to QUANTUM (15 % of correct answers when using the web and 9 % without). However, because of the important number of web answers that are unsupported (40 unsupported answers when using the web, 2 without), a strict evaluation cuts the performance by half, making the web version even worse than the no-web version (6 % of correct answers when using the web, 7 % without).

As for answer exactness, the proportion of exact answers over all the supported answers is about 16 % for both runs. We consider this a satisfactory result given the simplicity of the method we have chosen (single NPs only) and given that some of the errors are tagging errors, thus not related to the method itself.

The list-task run achieved an accuracy of 0.07, while our last year's best run achieved an accuracy of 0.15. Since all the answers that QUANTUM found this year are distinct and very few are unsupported, we attribute the performance decrease to the additional constraints on answer length (candidates are much smaller than 50 characters and some are inexact).

9 Conclusion

Even though we fine-tuned last year's version of QUANTUM by adding more question analysis patterns, by introducing function-specific weights in the

Run name	Web	# wrong	# unsupp'd	# inexact	# right	CWS
UdeMmainNoW	no	454	2	7	37	0.080
UdeMmainWeb	yes	425	40	6	29	0.057

Table 3: Detailed evaluation of the 2 main-task runs with their confidence-weighted score (CWS).

Run name	Web	Strict	Lenient
UdeMmainNoW	no	7 %	9 %
UdeMmainWeb	yes	6 %	15 %

Table 4: Strict (right answers only) and lenient (right, inexact and unsupported answers) evaluations of the 2 main-task runs. The score is the ratio over 500 questions.

computation of the candidates' scores and by decreasing the influence of WordNet to the benefit of Alembic, the increasing difficulty of the task resulted in a lower performance for this year. We feel that answer exactness can be achieved reasonably well by retrieving single-NP answers only, but candidate scoring, document support and no-answer questions are still challenging issues.

For the first time, we used the web as a source of answers. We derived from the questions the context in which we expected the answers to appear. Not taking into account answer exactness and document suppport, we found that 10 % of TREC-style questions can be correctly answered this way. We plan to go further in this direction by improving the generation of expected contexts and the validation of the candidates' semantic types.

Acknowledgments

We wish to thank Louis-Julien Guillemette whose programming skills gave QUANTUM access to the web. We also wish to express our greatest appreciation to the technical support staff of the DIRO department of the U. de Montréal. As Murphy's law would have predicted, the lab's file server crashed during the week of the competition. Without their fast and skillful response, we could not have made it on time this year.

This project was financially supported by the Bell University Laboratories (BUL) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] L. Plamondon and L. Kosseim. QUANTUM: A Function-Based Question Answering System. In *Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence (AI 2002)*, pages 281–292, Calgary, Canada, 2002.

[2] L. Plamondon, G. Lapalme, and L. Kosseim. The QUANTUM Question Answering System. In *Proceedings of The Tenth Text Retrieval Conference (TREC-X)*, pages 157–165, Gaithersburg, Maryland, 2001.

[3] S.E. Robertson and S. Walker. Okapi/Keenbow at TREC-8. In *Proceedings of TREC-8*, pages 151–162, Gaithersburg, Maryland, 1998.

[4] J. Aberdeen, J. Burger, D. Day, L. Hirschman, P. Robinson, and M. Vilain. MITRE: Description of the Alembic System as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference*, San Francisco, California, 1995. Morgan Kaufman Publishers.

[5] C.L.A. Clarke, G.V. Cormack, T.R. Lynam, C.M. Li, and G.L. McLearn. Web Reinforced Question Answering (MultiText Experiments for TREC 2001). In *Proceedings of The Tenth Text Retrieval Conference (TREC-X)*, pages 673–679, Gaithersburg, Maryland, 2001.

[6] E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data-Intensive Question Answering. In *Proceedings of The Tenth Text Retrieval Conference (TREC-X)*, pages 393–400, Gaithersburg, Maryland, 2001.