

QUANTUM: A Function-Based Question Answering System

Luc Plamondon¹ and Leila Kosseim^{2*}

¹ RALI, DIRO, Université de Montréal
CP 6128, Succ. Centre-Ville, Montréal (Québec) Canada, H3C 3J7
plamondl@iro.umontreal.ca

² Concordia University
1455 de Maisonneuve Blvd. West, Montréal (Québec) Canada, H3G 1M8
kosseim@cs.concordia.ca

Abstract In this paper, we describe our Question Answering (QA) system called QUANTUM. The goal of QUANTUM is to find the answer to a natural language question in a large document collection. QUANTUM relies on computational linguistics as well as information retrieval techniques. The system analyzes questions using shallow parsing techniques and regular expressions, then selects the appropriate extraction function. This extraction function is then applied to one-paragraph-long passages retrieved by the Okapi information retrieval system. The extraction process involves the Alembic named entity tagger and the WordNet semantic network to identify and score candidate answers. We designed QUANTUM according to the TREC-X QA track requirements; therefore, we use the TREC-X data set and tools to evaluate the overall system and each of its components.

1 Introduction

We describe here our Question Answering (QA) system called QUANTUM, which stands for QUEStion ANSwering Technology of the University of Montreal. The goal of QUANTUM is to find a short answer to a natural language question in a large document collection. The current version of QUANTUM addresses short, syntactically well-formed questions that require factual answers. By *factual answers*, we mean that they should be found directly in the document collection or using lexical semantics, as opposed to answers that would require world-knowledge, deduction or combination of facts. QUANTUM's answer to a question is a list of five ranked suggestions. Each suggestion is a 50-character snippet of a document in the collection, along with the source document number. The five suggestions are ranked from 1 to 5, the best suggestion being at rank 1 (see Fig. 1 for an example). QUANTUM also has the ability to detect that a question does not have an answer in the document collection. In that case, QUANTUM outputs an empty suggestion (with *NIL* as the document number) and ranks it

* Work performed while at the Université de Montréal.

according to its likelihood. Those features correspond to the TREC-X QA track requirements [1] where QUANTUM recently participated [2].

We shall introduce QUANTUM’s architecture and its function-based classification of questions. Then, we shall evaluate its overall performance as well as the performance of its components.

The TREC-X metric used to measure performance is called *Mean Reciprocal Rank (MRR)*. For each question, we compute a score that is the reciprocal of the rank at which the correct answer is found: the score is respectively 1, 1/2, 1/3, 1/4 or 1/5 if the correct answer is found in the suggestion at rank 1, 2, 3, 4 or 5. Of course, the score is 0 if the answer does not appear in any of the 5 suggestions. The average of this score over all questions gives the MRR.

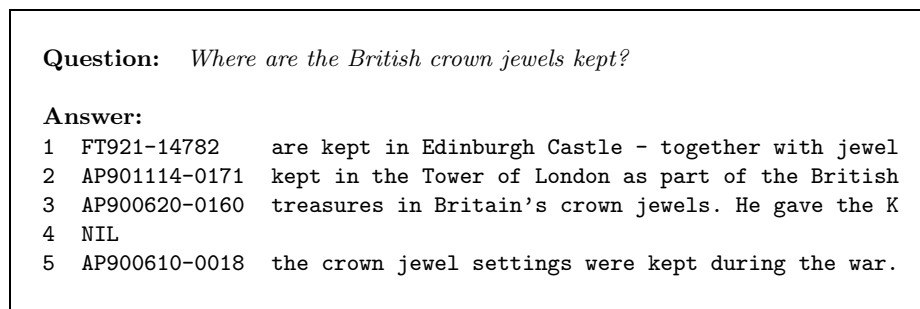


Figure 1. Example of a question and its corresponding QUANTUM output. Each of the five suggestions includes a rank, the number of the document from which the answer was extracted and a 50-character snippet of the document containing a candidate for the answer. A *NIL* document number means that QUANTUM suggests the answer is not present in the document collection. Here, the correct answer is found at rank 2.

2 Components of Questions and Answers

Before we describe QUANTUM, let us consider the question *How many people die from snakebite poisoning in the US per year?* (question # 302 of the TREC-9 QA track) and its answer. As shown in Fig. 2, 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. 3.3, 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

patterns shown in the table are given in Sect. 3.2 and 3.3). 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.

Table 1. Extraction functions, examples of TREC-X 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.

Function	Example of question and sample of answer patterns
$definition(\rho, \varphi)$	Q: <i>What is an atom?</i> ($\varphi = atom$) A: <hypernym of <i>atom</i> >, < <i>atom</i> or hyponym of <i>atom</i> > A: < <i>atom</i> or hyponym of <i>atom</i> > (<hypernym of <i>atom</i> >) A: < <i>atom</i> or hyponym of <i>atom</i> > is <hypernym of <i>atom</i> >
$specialization(\rho, \varphi)$	Q: <i>What metal has the highest melting point?</i> ($\varphi = metal$) A: <hyponym of <i>metal</i> >
$cardinality(\rho, \varphi)$	Q: <i>How many Great Lakes are there?</i> ($\varphi = Great\ Lakes$) A: <number> < <i>Great Lakes</i> or hyponym of <i>lake</i> >
$measure(\rho, \varphi)$	Q: <i>How much fiber should you have per day?</i> ($\varphi = fiber$) A: <number> <hyponym of <i>unit</i> > < <i>fiber</i> or hyponym of <i>fiber</i> > A: <number> <hyponym of <i>unit</i> > of < <i>fiber</i> or hyponym of <i>fiber</i> >
$attribute(\rho, \varphi)$	Q: <i>How far is it from Denver to Aspen?</i> ($\varphi = far$) A: Various patterns
$person(\rho)$	Q: <i>Who was the first woman to fly across the Pacific Ocean?</i> A: <PERSON named entity>
$time(\rho)$	Q: <i>When did Hawaii become a state?</i> A: <TIME named entity> A: <hyponym of <i>time-period</i> >
$location(\rho)$	Q: <i>Where is John Wayne Airport?</i> A: <LOCATION named entity> A: <hyponym of <i>location</i> >
$manner(\rho)$	Q: <i>How do you measure earthquakes?</i> A: Not implemented for TREC
$reason(\rho)$	Q: <i>Why does the moon turn orange?</i> A: Not implemented for TREC
$object(\rho)$	Default function A: <NP>

In most QA systems that use question classification [3] [4], a class is represented by a particular type of entity that the system is able to identify: toponyms, proper nouns, animals, weights, lengths, etc. In order to pair a question with an expected type of entity, one needs to anticipate all possible question forms that could focus on this type of entity. This introduces a supplemental difficulty, given the large number of possible reformulations of a question.

However, we performed a lexical and syntactic analysis of possible forms of English factual questions and found that the number of required search mechanisms is rather limited. By considering these mechanisms (our 11 functions) as classes, we facilitate the question classification task because the number of classes is small and because the classes are closely related to the syntax of questions. Even though the number of classes in such a function-based classification is smaller than in an entity-based classification, we can achieve the same level of precision by parameterizing our functions with the question focus when needed. The automated process of parameterizing a generic mechanism can suit questions about virtually any kind of entity, whereas an entity-based classification is limited to the entities it contains. In the worst cases, the chosen function f and parameter φ could lead to a generic, non-optimal search. Yet the correct answer can still be retrieved.

3.2 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. *Okapi* is an information retrieval engine that has the ability to return relevant paragraphs instead of whole documents [5]. 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 TREC-X factual questions are usually short noun phrases, 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 $person(\rho)$, $time(\rho)$ and $location(\rho)$, 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 [6], we currently use only PERSON, ORGANIZATION, LOCATION, DATE and TIME entities.

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

3.3 Extraction and Scoring of Candidates

Extraction. 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 (see Sect. 3.1). 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.

Scoring. To rank the candidates, each is assigned a score that is a combination of three partial scores: the extraction score, the passage score and the proximity score. The sum of these partial scores is used as the final score of a candidate.

Extraction score. The *extraction score* measures how confident we are in the search mechanism used. This score is awarded directly to a candidate by the extraction function called. Typically, we award a higher score to a candidate extracted using the named entity extractor or a hand-made pattern. A candidate extracted because it satisfies some WordNet hypernym/hyponym relation is given a lower score because of the higher risk of introducing noise (from polysemy for example).

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 directly use the score given to the passage during its retrieval by Okapi. 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.

Proximity score. The combination of the extraction score and the passage score favors candidates that have the type we looked for and that are related to the question context. We also give a *proximity score* to candidates contiguous to noun phrases that contain a question keyword (by *contiguous*, we mean that they are not separated by another noun phrase). If the answer is stated in a somewhat close syntactical reformulation of the question, we believe that the answer words should be close to the question keywords found in the text. We choose a relatively low proximity score to minimize its influence because it is a temporary substitute to a full syntactical parse of the passages that we wish to implement soon. At least, this score is helpful to break a tie between two candidates.

3.4 Candidate Expansion to 50 Characters

In the TREC-X QA track, answers are allowed to be snippets of up to 50 characters. To meet this, we expand each candidate by taking the 50-character document substring that is centered around it. Then, we cut off truncated words

at both ends, which allows us to shift the substring to the right or to the left so that the new 50-character string contains the maximum number of complete words. The purpose is to maximize the chances that the string contains the correct candidate in the unfortunate case where QUANTUM would have guessed wrong.

Candidate expansion takes place in conjunction with a redundancy elimination process. We begin by expanding our best candidate. Then, the second best candidate is expanded only if it does not appear in the first suggestion. The third candidate is expanded only if it does not appear in a previous suggestion, and so on until we have the desired number of suggestions.

3.5 No-Answer Questions

Until now, we have assumed that the answer of the question could be found in the document collection. However, this might not be the case: a *NIL* answer may thus be the correct answer indeed. To deal with this, we examine our candidates to determine whether a *NIL* answer should be amongst our 5 suggestions of answers and, if so, at what rank.

Since score scales differ from question to question (particularly when different extraction functions are used), we cannot use a unique score threshold below which we can say that a *NIL* answer is more likely than a low-score answer. Instead, we have used a threshold that depends on the score drop between two candidates and we have normalized it so that it can be applied to the candidates of any question.

Let a_i be the answer at rank i and δ_i^{i+j} be the score difference between a_i and its j^{th} successor a_{i+j} . We compute the normalized score drop Δ_i between a_i and a_{i+1} in the following manner:

$$\Delta_i = \frac{\delta_i^{i+1}}{\delta_i^{i+4}} = \frac{s_i - s_{i+1}}{s_i - s_{i+4}} \quad (1)$$

where s_i is the score of a_i . Our choice to normalize over a 5-rank score difference δ_i^{i+4} is arbitrary, though our experiments showed that the following observations still hold for normalization over different intervals.

We ran QUANTUM on the TREC-9 questions and kept all answers that were extracted (not only the 5 best). We then applied the official correction script to spot the rank r of the first correct answer (when found). We computed Δ_r to measure the normalized score difference between a correct answer and its successor, which was a wrong answer. We also computed the average Δ_i for any pair of answers of consecutive ranks. We found that the score drop between a correct answer and its successor is slightly higher than the average score drop between any pair of answers of consecutive ranks. Table 2 shows that this is true for different normalization intervals.

Therefore, QUANTUM applies the following algorithm to determine whether a *NIL* answer should be one of the 5 final suggestions. First, a_1 is considered correct because it has the highest score. The point is then to determine whether

Table 2. Normalized score drop Δ_r between a correct answer and its successor and Δ_i between any two answers of consecutive ranks. Results were obtained by running QUANTUM on the TREC-9 question set.

Normalization interval	Δ_r	Δ_i
δ_i^{i+4}	33 %	29 %
δ_i^{i+3}	40 %	35 %
δ_i^{i+2}	56 %	50 %

a_2 is more likely than a *NIL* answer at rank 2. If Δ_1 between a_1 and a_2 is high, then we have a supplemental hint that a_1 is correct and a_2 is incorrect, because we observed previously that a correct answer was usually followed by an important score drop (Table 2). Therefore, QUANTUM is confident enough in a_1 to say that if a_1 turns out to be an incorrect suggestion, then there is no satisfactory answer in the collection, so a *NIL* answer is inserted at rank 2 (and all other suggestions are shifted). Otherwise, if Δ_1 is low, then QUANTUM considers a_2 to be a good second choice; the algorithm is repeated, but this time assuming that a_2 is correct and investigating the likelihood of a *NIL* answer at rank 3. This process stops as soon as a *NIL* answer is inserted or after the insertion at rank 5 has been examined.

The Δ_r we computed previously between a correct answer and its successor is a lower bound for a threshold on Δ_i above which a *NIL* is inserted. We set this threshold Δ_t experimentally by creating a set of 400 questions in which we knew that 5 % of questions had no answer in the document collection (the remaining questions were from TREC-9). We then chose the threshold value Δ_t that maximized the overall MRR score (defined in Sect. 1) on this new question set. We obtained a maximum MRR score of 0.257 with $\Delta_t = 80$ %. The same experiment with 10 % of no-answer questions led to similar results.

4 System Evaluation

QUANTUM participated to the recent TREC-X QA track [1]. On the main track, QUANTUM’s best run received an MRR of 0.191¹, while the average of all systems was 0.234. We therefore wanted to evaluate each component of the system to see which modules were the most beneficial and which were the least.

We performed our detailed evaluation using the TREC-X question set, document collection and automatic correction tool. The different question sets we

¹ This result is the *strict evaluation* MRR, meaning that answers had to be supported by the document from which they were extracted. Moreover, answers were judged by human assessors. Though this is the official result, it cannot be directly compared with the results of the tests we present here since we used an automatic correction script, as we shall explain.

use in our experiments are all subsets of the 492 TREC-X questions. The document collection is made of about one million articles (3 gigabytes of text) from 6 newswires.

In our experiments, correction is done automatically by comparing the suggestions to a set of regular expressions. These patterns were compiled from the pooled 50-character strings that were submitted by all TREC-X QA systems and that were judged correct by human assessors from TREC. However, the automatic use of such patterns may not reflect the official TREC performance of a system. In some cases, the evaluation patterns are too restrictive. This is the case, for example, if a correct answer is in fact in the document collection but has not been encountered by the assessors, so that there are no pattern for this answer form. In other cases, the automatic evaluation can be too generous. This can happen, for example, when a pattern matches a string that has not been cited in a context related to the question. But regardless of these few cases of discrepancies, the use of the automatic patterns allows us to evaluate different experiments quickly and consistently.

4.1 Evaluation of the Classification Module

Table 3 shows that 88 % of the 492 TREC-X questions were correctly analyzed by our 40 question patterns, so that the correct extraction function and the correct focus were used to find candidate answers. The extraction functions that suffered the most from classification errors are *definition*(ρ, φ) (24 % of *definition* questions were assigned an erroneous function and/or focus) and *unknown*(ρ) (12 % were considered known and assigned an erroneous function).

Table 3. Misclassified questions per extraction function in the TREC-X question set. Note that *reason*, *manner* and *object* questions are tagged as *unknown* in the current version of QUANTUM.

Function	Number of questions		Analysis by QUANTUM			
			Correct funct. Correct focus	Correct funct. Wrong focus	Wrong funct.	
<i>definition</i> (ρ, φ)	140	(29 %)	107 (76 %)	11 (8 %)	22 (16 %)	
<i>specialization</i> (ρ, φ)	194	(40 %)	177 (91 %)	7 (4 %)	10 (5 %)	
<i>cardinality</i> (ρ, φ)	13	(3 %)	12 (92 %)	0 (0 %)	1 (8 %)	
<i>measure</i> (ρ, φ)	1	(0 %)	1 (100 %)	0 (0 %)	0 (0 %)	
<i>attribute</i> (ρ, φ)	22	(4 %)	20 (91 %)	0 (0 %)	2 (9 %)	
<i>person</i> (ρ)	43	(9 %)	40 (93 %)	—	3 (7 %)	
<i>time</i> (ρ)	26	(5 %)	26 (100 %)	—	0 (0 %)	
<i>location</i> (ρ)	27	(5 %)	27 (100 %)	—	0 (0 %)	
<i>unknown</i> (ρ)	26	(5 %)	23 (88 %)	—	3 (12 %)	
Total	492	(100 %)	433 (88 %)	18 (4 %)	41 (8 %)	

About half of the classification errors (47 %) are due to unexpected question forms to which our set of patterns would not apply. By designing new patterns, these errors can easily be reduced. About a third of the classification errors (29 %) are due to our part-of-speech tagger. This was to be expected, as our tagger is probabilistic and has not been trained on a corpus of questions; therefore, it is not optimal to capture the particular syntax of interrogative sentences. Our NP-chunker based on part-of-speech tag patterns caused 15 % of the errors and their correction would require to disambiguate some tag patterns using more complex techniques. The remaining 8 % is due to other sources of error.

4.2 Evaluation of the Extraction Module

In order to evaluate the performance of the tools used by the extraction module, we ran several experiments with and without them. Table 4 shows that the complete QUANTUM system, but without *NIL* insertion, can achieve a maximal MRR score of 0.223 when run on questions that it can correctly classify and that are not no-answer questions. Without WordNet, the performance drops by 7 % to 0.207. It is interesting to note that questions answered by the *time*(ρ) or *person*(ρ) extraction function benefit from the removal of WordNet. We believe that this is due to noise introduced by WordNet. The most valuable module added to the core of QUANTUM seems to be Alembic. Without it, the performance drops by 22 % to 0.175. The functions that rely heavily on this named entity extractor, such as *location*(ρ), *person*(ρ) and *time*(ρ), have their MRR reduced by one half without it. Finally, without hand-made regular expression patterns for answer extraction, the overall performance drops by a small 2 % to 0.218. This is because very few patterns were used, except for the extraction of definitions, where they do seem to be useful.

4.3 Evaluation of the *NIL* Insertion Module

The inclusion of no-answer questions in the question set and the insertion of *NIL* answers based on the score drop between candidate answers lead to an overall MRR of 0.199, compared to 0.223 without no-answer questions and *NIL* insertion. We believe that this drop in performance is due to the supplemental difficulties that no-answer detection introduces, as well as the poor performance of the *NIL* insertion module (only 5 out of 49 no-answer questions were correctly answered).

5 Discussion and Conclusion

The best-scoring systems in TREC-X have used one of 3 approaches: some systems rely solely on information retrieval techniques [7], some rely heavily on world-knowledge [8], and some, like QUANTUM, rely on both information retrieval and general computational linguistics techniques. In addition, several systems use redundancy of answers found in different sources, namely the Web:

Table 4. Overall MRR and MRR per extraction function for different versions of QUANTUM on the TREC-X question set. The *# of questions* is the number of questions correctly analyzed and for which the correct answer is not *NIL*. The versions of QUANTUM tested here do not perform *NIL* answer insertion.

Function	# of quest.	MRR			
		QUANTUM complete	QUANTUM w/o WordNet	QUANTUM w/o Alembic	QUANTUM w/o regexps
<i>definition</i> (ρ, φ)	113	0.179	0.159	0.181	0.158
<i>specialization</i> (ρ, φ)	153	0.205	0.170	0.175	0.205
<i>cardinality</i> (ρ, φ)	12	0.096	0.086	0.096	0.096
<i>measure</i> (ρ, φ)	1	0.000	0.000	0.000	0.000
<i>attribute</i> (ρ, φ)	18	0.019	0.056	0.074	0.074
<i>person</i> (ρ)	38	0.348	0.375	0.205	0.346
<i>time</i> (ρ)	24	0.411	0.418	0.206	0.411
<i>location</i> (ρ)	25	0.451	0.415	0.188	0.418
<i>unknown</i> (ρ)	21	0.129	0.129	0.129	0.148
Total	405	0.223	0.207	0.175	0.218

some systems use redundancy as re-enforcement to adjust the score of candidates found in the document collection [7], while others rely solely on the Web without searching the collection [9]. As Clarke et al. report [7], Web-reinforcement can yield a substantial increase in performance with the TREC data set.

In comparison with other approaches, our approach to QA is closer in essence to that of Harabagiu et al. [3]. It uses general computational linguistics techniques such as named entity extraction and lexical semantics without consulting external sources for answers. Though the performance of QUANTUM at the recent TREC-X conference [10] is rather average, as mentioned in Sect. 4, we believe that the approach we have chosen is more adaptable to other QA applications than the approaches taken by other (better scoring) systems. Indeed, it can be used to answer general-knowledge questions such as those used in TREC-X or to answer domain-specific questions such as those received by a company’s customer support service, where the sources of answers are limited and redundancy cannot be exploited.

In this paper, we have described QUANTUM, a QA system that extracts short answers to fact-based questions from a large document collection. In the system, we used a classification of questions made of extraction functions and we tried to avoid the use of hand-made answer-extraction patterns to remain generic. Different experiments with the TREC-X material showed that the Alembic named entity extractor is very helpful for extracting answers. However, the use of WordNet, as currently used in QUANTUM, should be tuned because it seems to induce almost as much noise as benefit.

5.1 Acknowledgments

We wish to thank Guy Lapalme, without whom this project would not have started. We also wish to thank Sylvain Laganière and Luc Bélanger for their help, the Mitre Corporation for making Alembic Workbench available for research purposes and Ken Litkowsky (CL Research) for providing us with his TREC-X answer patterns.

This project was financially supported by a scholarship from the Québec *Fonds pour la formation de chercheurs et l'aide à la recherche* (FCAR), the Bell University Laboratories (BUL) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

- [1] E.M. Voorhees. Overview of the TREC 2001 Question Answering Track. In *Notebook Proceedings of The Tenth Text REtrieval Conference (TREC-X)*, pages 157–165, Gaithersburg, Maryland, 2001.
- [2] L. Plamondon, L. Kosseim, and G. Lapalme. The QUANTUM Question Answering System. In *Notebook Proceedings of The Tenth Text REtrieval Conference (TREC-X)*, pages 571–577, Gaithersburg, Maryland, 2001.
- [3] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu. FALCON: Boosting Knowledge for Answer Engines. In *Proceedings of The Ninth Text REtrieval Conference (TREC-9)*, pages 479–488, 2000.
- [4] O. Ferret, B. Grau, M. Hurault-Plantet, G. Illouz, C. Jacquemin, N. Masson, and P. Lecuyer. QALC - The Question-Answering System of LIMSI-CNRS. In *Proceedings of The Ninth Text REtrieval Conference (TREC-9)*, pages 325–334, 2000.
- [5] S.E. Robertson and S. Walker. Okapi/Keenbow at TREC-8. In *Proceedings of The Eighth Text REtrieval Conference (TREC-8)*, pages 151–162, Gaithersburg, Maryland, 1998.
- [6] 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 (MUC-6)*, San Francisco, 1995. Morgan Kaufman Publishers.
- [7] 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 *Notebook Proceedings of The Tenth Text REtrieval Conference (TREC-X)*, pages 620–626, Gaithersburg, Maryland, 2001.
- [8] E. Hovy, U. Hermjakob, and C.-Y. Lin. The Use of External Knowledge in Factoid QA. In *Notebook Proceedings of The Tenth Text REtrieval Conference (TREC-X)*, pages 166–174, Gaithersburg, Maryland, 2001.
- [9] E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data-Intensive Question Answering. In *Notebook Proceedings of The Tenth Text REtrieval Conference (TREC-X)*, pages 183–189, Gaithersburg, Maryland, 2001.
- [10] NIST. *Notebook Proceedings of The Tenth Text REtrieval Conference (TREC-X)*, Gaithersburg, Maryland, 2001.