Proceedings of the Australasian Language Technology Workshop 2007, pages 92-99

Question Prediction Language Model

Luiz Augusto Pizzato and **Diego Mollá**
Centre for Language Technology
Macquarie University
Sydney, Australia
{pizzato, diego}@ics.mq.edu.au

Abstract

This paper proposes the use of a language representation that specifies the relationship between terms of a sentence using question words. The proposed representation is tailored to help the search for documents containing an answer for a natural language question. This study presents the construction of this language model, the framework where it is used, and its evaluation.

1 Introduction

Although Information Retrieval (IR) can be helped by NLP techniques such as named entity (NE) recognition, phrase extraction and syntax parsing (Strzalkowski, 1999), they are not generally used due to their high complexity. One such task that people can perform somewhat easily whilst still being hard for computers is the answering of factoid questions based on textual content. The Question Answering (QA) Track of TREC (Voorhees, 2005) focuses on answering questions using the AQUAINT corpus (Graff, 2002), which contains 375 million words from three different sources of newswire data: Xinhua News Service (XIE) from People's Republic of China, the New York Times News Service (NYT), and the Associated Press Worldstream News Service (APW).

For the QA task, not only it is important to find an answer in a document, but also to find the documents that might contain the answer in the first place. Most QA systems take the approach of using off-the-shelf IR systems to return a list of documents that may contain an answer, and then processing the list of documents to look for the required answer. Normally the processing time for every question in these systems is long because of the sheer amount of work that is required after the list of document is returned.

Many QA systems focus on the input and output of IR systems. For example, Dumais et al. (2002) perform a passive-to-active voice transformation of the question, in an attempt to bring the IR query closer to the document it is expected to retrieve. Some IR work focuses on improving QA by passage retrieval re-ranking using word overlap measures. For instance, Tellex et al. (2003) compare a group of passage retrieval techniques and conclude that those that apply density-based metrics[1] are the most suitable to be used for QA.

Some work has been done on IR models that specifically aid the QA task. The work of Monz (2004) defines a weighting scheme that takes into consideration the distance of the query terms. Murdock and Croft (2004) propose a translation language model that defines the likelihood of the question being the translation of a certain document. Tiedemann (2005) uses a multi-layer index containing more linguistic oriented information and a genetic learning algorithm to determine the best parameters for querying those indexes when applied for the QA task. Tiedemann argues that since question answering is an all-natural language task, linguistic oriented IR will help finding better documents for QA.

In this paper we propose a language representa-

[1] Ranking of passages based on the number of query words and the proximity between them.

tion that when used in the IR stage of a question answering system improves its results. As a consequence it helps to reduce the processing time due to a better retrieval set and because it has the capacity of giving answer cues.

This paper is divided into five sections. The next section presents the Question Predication Language Model and some of its features. Section 3 introduces how the the model is used and how the necessary resources for its usage were built. Section 4 describes some experiments and present some preliminary results. Section 5 presents the concluding remarks and future work.

2 Question Prediction Language Model

We describe a language model that focuses on extracting a simple semantic representation of an English text that can be easily stored in digital databases and processed by Information Retrieval (IR) tools. We focus on extracting a particular kind of semantic that help us to find the location of a text that has some likelihood of answering a question. The model and its semantic are defined as Question Prediction (QP).

The Question Prediction Language Model (QPLM) represents sentences by specifying the semantic relationship among its components using question words. In this way, we focus on dividing the problem of representing a large sentence into small questions that could be asked about its components. In other words, we represent the relationship among key words of a sentence as short questions. For instance, the sentence "*Jack eats ham*" could be represented by the following two triples: $Who(eat, Jack)$ and $What(eat, ham)$. Using this model it is possible to answer short questions that focus on relations existent inside a sentence context, such as "*Who eats ham?*" and "*What does Jack eat?*".

The QPLM represents sentences as semantic relations expressed by triples $q(w, a)$ where q is a question word, w is the word that concerns the question word q and a is the word that answers the relation q about w. For instance the relation $Who(eat, Jack)$ tells us that the person who eats is Jack. The representation of our semantic relations as triples $Q(w, a)$ is important because it allows the representation of

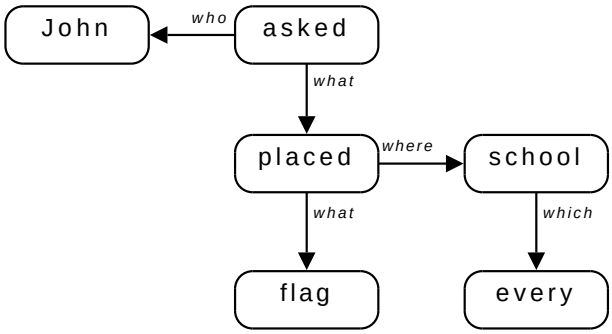

Figure 1: Graph Representation

sentences as directed graphs of semantic relations. This representation has the capacity of generating questions about the sentence being analysed. Figure 1 shows such a representation of the sentence: "*John asked that a flag be placed in every school*".

Having the sentence of Figure 1 and removing a possible answer a from any relation triple, it is possible to formulate a complete question about this sentence that would require a as an answer. For instance, we can observe that removing the node $John$ we obtain the question "*Who asked for a flag to be placed in every school?*" where Who was extracted from the triple $Who(ask, John)$. The same is valid for other relations, such as removing word $school$ to obtain question "*Where did John asked for a flag to be placed?*". The name Question Prediction for this model is due to its capability of generating questions regarding the sentence that has been modeled.

In this section, we have shown how our model represents the semantic information. In the next section we focus on the implementation of QPLM and its usage.

3 Building and using QPLM

As observed in Figure 2, a training set of QPLM triples was created using mapping rules from a corpus of semantic role labels. Using a syntactic parser and a NE recognizer with our training set, we were able to learn pattern rules that we further applied in the processing of the AQUAINT corpus.

PropBank (Palmer et al., 2005) is a corpus with annotated predicate-argument relations from the same newswired source of information as the Penn Treebank[2]. We used PropBank as our starting point

[2] http://www.cis.upenn.edu/ treebank

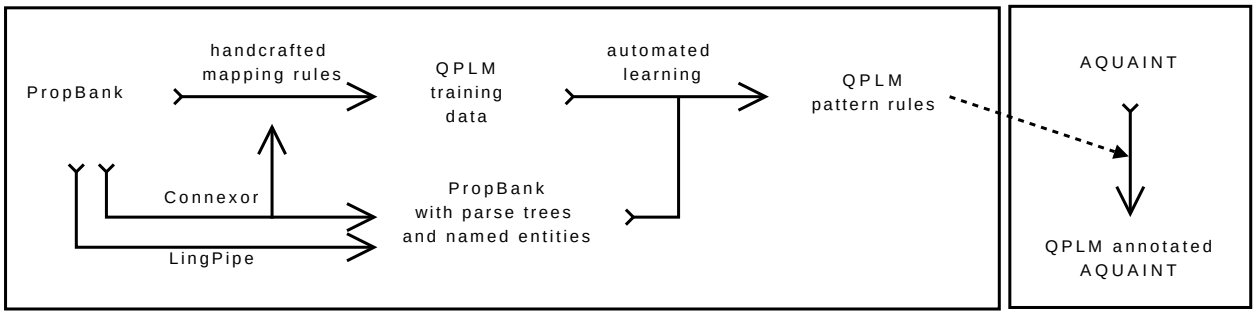

Figure 2: Creation and usage of pattern rules.

because it comprises the same textual style, and the predicate-argument relations (also referred to as semantic roles) can be mapped to QPLM triples.

We studied the possibility of using semantic role labeling tools to perform the semantic annotation, however our experiments using these tools showed us that they have not yet achieved a reasonable speed performance. For instance, the SwiRL semantic role labeling system[3] would take a couple of years to fully process the AQUAINT corpus. In contrast, our system takes a couple of days if all the necessary information is already at hand; adding the time required for syntactic parsing and NE recognition, the total processing period is not longer than two weeks.

3.1 Training corpus

PropBank is processed through a set of mapping rules from the predicate-argument relations to QPLM. Using a PropBank map as our training data gives us the benefit of a large training set, but at the same time it will only create relations that are present in PropBank, therefore excluding some relations that we wish to include. For instance, relations that do not involve any action, such as the ownership relation in $(Whose(car, Maria))$ and the quantity relation in $(HowMany(country, twenty)))$, among others.

PropBank defines relations between predicate and arguments without properly defining their meaning. On the other hand, it does keep a format where the argument number 0 represents the agent acting upon something and argument number 1 represents patients or themes. PropBank was manually annotated according to the PropBank Marking Guidelines (Babko-Malaya, October 2006). The guidelines represent an effort to build a consistent set of relations, however a closer look at the corpus shows that consistency is a hard task to achieve, particularly with the vaguely defined arguments number 3 onwards. For those cases the inclusion of a function tag proved to be useful[4].

Observing how arguments and predicates relate to each other, we created a set of rules mapping from argument-predicate relations to the QPLM. The basic differences between both models is that the QPLM triple contains a label representing a more specific semantic relation, and that it associates only the head of the linked phrases. For instance, the sentence "*The retired professor received a lifetime achievement award*" is represented as:

(1) **Semantic Roles**: [The retired professor]^{ARG0} [received]^{pred} [a lifetime achievement award]^{ARG1}.

(2) **QPLM**: Who(receive, professor), What(receive, award)

As can be observed in (1), semantic role labeling does not provide information about which is the main term (normally the head of a phrase) of each argument, while in (2), QPLM represents relations between the phrase heads. In order to find the phrase head, we applied a syntactic parser (Connexor[5]) to PropBank sentences. However, the phrase heads are not always clearly defined (particularly when the syntactic parse tree is broken due to problems in the parser) creating an extra difficulty for the mapping process. When a syntactic path cannot be found between predicates and any of the words from the argument, we then try to find the head of the phrase

[3] http://swirl-parser.sourceforge.net/

[4] A function tag is information attached to the arguments representing relations such as negation, location, time and direction.

[5] http://www.connexor.com

by syntactically parsing the phrase by itself. If this also fails to provide us with a head, we simply use the first available non-stopword if possible.

The stage of finding the related phrases heads showed to be quite important, not only because we would be defining which words relate to each other, but also because if a broken parse tree is found, no rules could be learnt from the resulting QPLM triple. An analysis of the data showed us that 68% of the QPLM triples derived from PropBank were generated from an unbroken parse, while the rest used some of the other methods.

We understand that even though our model has similarities with Semantic Role Labeling, we are taking a step further in the sense of semantic representation. QPLM has a finer semantic representation meaning that a predicate argument relation in PropBank might have different representations in QPLM. Our mapping rules takes into consideration not only the number of the argument but also the predicate involved and the POS or NE of the related words.

Even though we cover different aspects of PropBank in our mapping, we observed that many predicates hold different meanings for the same arguments which creates a problem for our mapping strategy. This problem was not fixed because of the prohibitive amount of work needed to manually mark all the different meanings for the same predicate in different sentences. In these cases, where the same predicates and the same argument represent different semantics according to the QPLM, we chose the one most representative for the set of sentences using that predicate and argument. For instance, the argument number 3 of predicate *spend* for the majority of the cases represents a quantity of money that was spent (a HowMuch label), however we have one case where the argument is *cash* (a What label). This type of mapping compromises the accuracy of our conversion, however a randomly selected set of 40 documents was manually evaluated showing that nearly 90% of the QPLM triples were correctly converted.

After the mapping was finalized we obtained a training set of rules with 60,636 rules, and 39 types of semantic relations (Table 1).

aboutwhat	do	outofwhat
adv	forwhat	overwhat
afterwhat	fromwhat	subj
againstwhat	how	towhat
aroundwhat	howlong	towhom
aswhat	howmuch	underwhat
atwhat	howold	what
behindwhat	intowhat	when
belowwhat	inwhat	where
beneathwhat	likewhat	who
betweenwhat	obj	whom
beyondwhat	ofwhat	why
bywhat	onwhat	withwhat

Table 1: QPLM Semantic Relations

Original: John kicked the ball bought by Susan.
QPLM: Who(kick, John), What(kick, ball), What(buy, ball), Who(buy, Susan)
Parse Tree: $John_{np}{}^{subj} \rightarrow kick_{va} \leftarrow^{obj} ball_{nn} \leftarrow^{det} the_{det}$ $ball_{nn} \leftarrow^{mod} buy_{vp} \leftarrow^{agt} by_{prep} \leftarrow^{pcomp} Susan_{np}$
Named Entities: <ENAMEX Type=NAME> John </ENAMEX> kicked the ball bought by <ENAMEX Type=NAME> Susan </ENAMEX>.

Table 2: Training Files

3.2 Rule learning

The PropBank corpus, after being automatically converted to QPLM triples, is used to learn the rules that are used to find the QPLM information of plain text. The QPLM annotation relies on the output of a syntactic parser and of a named-entity recognizer for its annotation and for the rule learning process. We are currently using Connexor for syntax parsing and LingPipe[6] to recognize NEs. Our semantic model uses pattern rules (PRules) created from the representation of the same sentence as syntactic parse trees, MUC style named entity, and a list of QPLM triples. Table 2 presents the different information that we use for training.

Having these representations at hand, a set of rules is learned using the following process (see Figure 3 for an example):

1. replace the part of speech information with the respective named entity category in the syntactic parse tree;

[6] http://www.alias-i.com/lingpipe/

2. identify leaf-to-root links along the combined syntactic and named-entity (S+NE) path between w and a for every triple $Q(w, a)$;

3. for the existing S+NE paths, replace w and a by a marker in both the triples and the paths, registering those as pattern rules (PRule);

   repeat steps 2 to 3 for all triples and documents;

4. combine all PRules found, calculate their frequency of occurrence and group them by common triples. It is important to note that if we have a sentence such as "*Jack eats*", we would have a frequency of two ($2\times$) for the pattern ${a_{person}}^{subj} \rightarrow w_{va}$.

1. ${John_{\mathbf{person}}}^{subj} \rightarrow kick_{va}$
2. $Who(kick, John) :\ {John_{person}}^{subj} \rightarrow kick_{va}$
3. $Who(w, a) :\ {a_{person}}^{subj} \rightarrow w_{va}$
4. $Who(w, a) :$
   - $1\times :\ {a_{person}}^{subj} \rightarrow w_{va}$
   - $1\times :\ {a_{person}}^{pcomp} \rightarrow {by_{prep}}^{agt} \rightarrow w_{vp}$

Figure 3: Process example

After computing all the training files we would have a resulting PRule file containing all possible S+NE paths that can generate the manually defined triples. If an S+NE path could not be found then a PRule cannot be generated and the current training triple is skipped.

3.3 Applying QPLM

Using the training corpus described above, we found all the PRules needed in order to generate the semantic triples when having an S+NE representation. The rules are grouped by QPLM triples, having their S+NE paths attached with a frequency value. This frequency value represents how many times an S+NE path was used to generated a PRule in the training corpus.

To convert S+NE files into QPLM, we start by applying those PRules that have the highest frequency values. These PRules are believed to be the most significant ones. Also it is important to observe that if an S+NE path generates different QPLM triples, we only need to apply the one with the higher frequency. For instance, if the pattern ${w_{person}}^{subj} \rightarrow a_{va}$ is associated with the triple $Who(w, a)$ with frequency of 8 and with the triple $Where(w, a)$ with a frequency of 2, the S+NE path will only generate the Who triple. Because frequency is the decisive factor, in the previous example we have 20% of chance of wrongly assigning an incorrect semantic label.

We observed that more precise PRules could be created taking into account that some verbs constantly generate a different QPLM triple for the same S+NE path. These new PRules (which we refer to as FW) are defined with a fixed w becoming less frequent but at the same time more precise. The precision of FW rules combined with the generality of the previous ones (which we refer to as GN) assure us that we have a correct analysis of a known verb as well as fair guess of an unseen one. To ensure that known verbs are evaluated first by the more precise FW rules, we assign a much higher weight to those rules than GN ones. An evaluation using the combination of both types of rules has shown us that assigning a weight 800 times higher to FW than to GN gives us the best results.

We also observed that due to the large amount of learnt PRules, the process for creating the QPLM was slow. In order to improve the speed performance of the process, we decided to compromise our system precision and recall by removing the least important rules, i.e. those with a frequency equal to one. The lower number of PRules caused a decrease of recall which is more noticeable when taking into account the FW rules. Even though we experienced a decrease of precision, removing low frequent PRules causes the removal of abnormal PRules that were generated by parsing errors.

In the next section we describe the environment where QPLM was applied, followed by some experimental results.

4 Evaluation

It is possible to evaluate our model implementation on how well it performs the task of assigning the correct semantic labels to a certain text. However because the model was designed so it would improve the IR stages of a QA system, we believe that the most significant evaluation at this point is in terms

of how well it helps us solving this specific problem.

Since we have not yet implemented lexical semantic substitution or any other IR techniques such as stemming or lemmatization, a comparison with a full-fledged state-of-the-art IR system is not relevant. The lack of these techniques makes it somewhat harder to boost the confidence on single sentences or windows of text if the proper context is not recorded. However, we have confidence that the model can help to provide cues of possible answers by substituting the partial match between a question word and document representation. For instance, if the question "*Where will a flag be placed?*" is presented and the sentence in Figure 1 is considered, a partial match *school* can be considered as a possible answer.

4.1 The IR model comparison

We have compared our QPLM triples with bag-of-words (unigram) and syntactic dependency triples. In all three cases the indexing and retrieval methods were the same, only the indexed units were different. We have implemented an IR framework that can hold relational information such as n-grams, syntactic, semantic role label and QPLM. Our framework is implemented so that it supports fast indexing (hundreds of documents per second in a low-end desktop machine) and it retrieves using a vector space model. The framework allows distributed indexing and retrieval under other models but they are not yet implemented.

During the development and test phases of our framework, we have implemented different vector space models of retrieval. We have implemented the unigram model, along with syntactic relations, semantic role labeling and QPLM.

The unigram model associates words with documents as well as it adds the position of them within the document. The inclusion of position information allows the use of a ranking function based on proximity. We also implemented a syntactic model using the output of Connexor. The model associated words with documents as well as words with their respective heads and/or modifiers. Since we are using a vector space model, computing TF and IDF over this type of relation has a different meaning. TF for full matching triples would be how many of them with the same $Syntactic - Relation$, $head$ and $modifier$ are found, while TF could also mean partial matches where one or two elements are not needed to match. IDF in this setup would be similar to TF but with the scope of the whole document set.

In the semantic role labeling model, a complete match of predicate and argument is not expected; because of this we only consider partial matches (if a word appears as the correct argument or predicate). However we expect a larger weight for a document when all the words properly match. In this model IDF and TF are calculated by taking into account the words found in the right context.

In the QPLM we have a very similar model to the syntactic relation one. However in QPLM not all the words will relate to each other, causing a large number of words to be missing in the final representation. To overcome this problem, we compute IDF and TF as the syntactic relations and we also add the TF/IDF weights from an unigram model. Because QPLM relations are much less frequent, when a match is found they will have higher weights than unigrams. Unigrams and QPLM are combined so that we do not discard relevant documents when they contain important keywords that are missing in the QPLM representation.

4.2 Evaluation over IR and QA

We have shown in the previous sections that the QPLM analysis relies on a syntactic parser and on a named-entity recognizer, in order to build a training set used to look for pattern rules and then to analyse text files into this model. We have not analysed the correlation among the performance of the syntactic parser nor the named entity recognizer, however we observed that our model has problems learning rules when these components offer poor results. As explained previously in section 3.1, if we cannot find a rule connecting two nodes into the same parse tree, we cannot learn a conversion rule. These cases account for 42,609 out of 135,537 rules, reducing in practical matters our training set to only 68% of its original size. Many of these cases are due to broken parse trees returned by Connexor. We have not yet experimented with different parsers, however a possible outcome of such experiment might be that having a broken structure and therefore losing the training instance is more desirable than having the full parse, but with the wrong dependencies.

We have also filtered out the pattern rules that have the same S+NE path but are not the most frequent one regarding their QPLM triple. By doing this we discard 12% of all the rules (20% of GN and 4% of FW). We do not use the rules when their frequency values are equal to one, this will cause an extra drop in the number of rules used to 44%. As expected the removal of low frequency rules have a stronger impact on FW rules than in GN rules (54% and 33% respectively).

This information is important because we can then predict what the upper limit of our performance is when measuring the recall using the set of rules we built as a training and testing set. According to the values presented, using all the rules with a frequency of 2 or more we have an upper limit of recall of 38%. A ten-fold evaluation over the training set has given us a value of 24% for recall.

When comparing the PropBank mapped files with the files analysed by our technique, it is possible to observe that the amount of QPLM triples in our semantic analysis is much larger than the ones mapped from PropBank. The reason is that PropBank only marks certain predicates in a sentence, while QPLM also provides relation among other verbs and other sentence words. Because of this we performed a manual evaluation of the precision of our technique over a set of 20 randomly selected documents and we found that 50% of the relations can be seen as correct. We also observed that many of the relations that were wrongly generated were due to some errouneous S+NE path. Filtering out this wrong pattern from the rule file will improve our precision. The important fact is that even though our performance over the analysis of the QPLM does not appear to be very high, the generated rules show to be very useful when applied to IR and the QA task.

We have retrieved one hundred documents for the set of 1449 questions of the TREC 2004, 2005 and 2006 QA track (Voorhees, 2005; Voorhees and Dang, 2006) and verified the existence of the answer string in each of these documents. We have performed this retrieval process for the unigram, syntactic relations and QPLM models. Due to data storage constraints at this moment we only have the results for the XIE and APW newswire corpus.

The comparison between the three models shows that we can obtain better documents to answer natural language questions if we take into consideration the same linguistic relations in the question and in the terms present in the documents. We measure our results using coverage and redundancy measures (Roberts and Gaizauskas, 2004). Coverage tells us how much of the question set we can answer using the top-N documents, while redundancy tells us how many documents per question contain an answer. These results are presented in Table 3.

Coverage

n docs.	5	10	25	50	100
UNI	0.2227	0.2718	0.3246	0.3591	0.3885
SYN	0.2307	0.2752	0.3325	0.3691	0.3964
QPLM	0.2310	0.2787	0.3370	0.3757	0.4040

Redundancy

n docs.	5	10	25	50	100
UNI	0.5421	1.0062	2.2279	4.0363	7.2158
SYN	0.5715	1.0535	2.3080	4.1091	7.2686
QPLM	0.5729	1.0584	2.3567	4.2486	7.6405

Table 3: Results for APW and XIE

As we observe in Table 3, for a large collection of documents and questions our system performs consistently better than unigram and syntactic relations. We have performed a paired t-test for statistic significance using the results of the individual questions for QPLM and unigrams showing that there is a 99% percent of chance that the improvement is not random for the results in the APW corpus. However, a paired t-test did not reject the null hypothesis in the test performed with the XIE corpus. This may be an indication that the XIE Newswire Corpus is written with a linguistic style that our system has not been able to take advantage of. Perhaps it strongly differs from the style present in our training set (PropBank) causing our rules not to be successfully used. Further work is needed to understand the main differences among these corpora. By understanding this we might find ways to adjust our system towards different textual styles.

Even though coverage and redundancy are good measures for evaluating a retrieval set for QA, we have observed that these measurements do not always relate to each other (Pizzato et al., 2006). For this reason we have applied the retrieval set to a QA system in order to observe if it does help to improve its results. Using the retrieval sets generated by the different models in the AnswerFinder QA system

(van Zaanen et al., 2007) showed us that the QPLM performed 25% better than the unigram model and 9.3% better than the syntactic model. Even though AnswerFinder is not among the best performing QA systems, it does give us some insight on what a retrieval set should contain.

5 Concluding Remarks

In this paper we have presented a semantic model that represents the relations among sentence components by labeling them with question words. This model was built to assist the task of question answering, particularly at the IR stages. We understand that by providing documents that are better suited towards finding an answer for a natural language question, QA system would not only return better answers but also become faster.

The work presented here shows that the QPLM can be used effectively in IR frameworks, and a comparison with the unigram and syntactic model demonstrates that we are able to improve the overall IR results. We have already implemented the predicate-argument model in our IR framework and we plan to compare it with QPLM. Because the current semantic role labeling systems are impractically slow when applied to large corpora, the comparison will be done using a reduced number of documents.

In this work, we focused on the single impact of our technique on the retrieval stages of a QA system. In future work we will include different retrieval methods in our IR framework to enable a valid comparison with state-of-the-art IR systems. We also plan to manually study the PRules so as to identify the ones causing some drops in the precision and recall of our model, and to construct an automatic method that would help this process.

As explained, we had data storage constraints which made the evaluation more difficult. As future work we plan to distribute the retrieval process and to perform evaluations with the whole AQUAINT corpus and with the NYT documents. We also intend to evaluate the impact that different retrieval sets have in a broader range of QA systems.

References

O. Babko-Malaya. October 2006. Propbank annotation guidelines.

S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. 2002. Web question answering: is more always better? In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–298, Tampere, Finland. ACM Press.

D. Graff. 2002. *The AQUAINT Corpus of English News Text*. Linguistic Data Consortium, Philadelphia.

C. Monz. 2004. Minimal span weighting retrieval for question answering. In *Proceedings of the SIGIR-2004 Workshop on Information Retrieval For Question Answering (IR4QA)*, Sheffield, UK, July.

V. Murdock and W.B. Croft. 2004. Simple translation models for sentence retrieval in factoid question answering. In *Proceedings of the SIGIR-2004 Workshop on Information Retrieval For Question Answering (IR4QA)*, Sheffield, UK, July.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguist*, 31(1):71–106.

L. A. Pizzato, D. Molla, and C. Paris. 2006. Pseudo relevance feedback using named entities for question answering. In *Proceedings of the Australasian Language Technology Workshop 2006.*, Sydney.

I. Roberts and R. J. Gaizauskas. 2004. Evaluating passage retrieval approaches for question answering. In *ECIR*, volume 2997 of *Lecture Notes in Computer Science*, pages 72–84. Springer.

T. Strzalkowski. 1999. *Natural Language Information Retrieval*. Kluwer Academic Publishers, Norwell, MA, USA.

S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 41–47, Toronto. ACM Press.

J. Tiedemann. 2005. Optimizing information retrieval in question answering using syntactic annotation. In *Proceedings of RANLP 2005*, pages 540–546, Borovets, Bulgaria.

M. van Zaanen, D. Molla, and L. A. Pizzato. 2007. Answerfinder at trec 2006. In *The Fifteenth Text REtrieval Conference (TREC 2006)*.

E. M. Voorhees and H. T. Dang. 2006. Overview of the TREC 2005 question answering track. In *Text REtrieval Conference*.

E. M. Voorhees. 2005. Overview of the TREC 2004 question answering track. In *Text REtrieval Conference*.