# Effective and Efficient Structured Retrieval

Le Zhao and Jamie Callan
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
{ lezhao, callan }@cs.cmu.edu

## ABSTRACT

Search engines that support structured documents typically support structure created by the author (e.g., title, section), and may also support structure added by an annotation process (e.g., part of speech, named entity, semantic role). Exploiting such structure can be difficult. Query structure may fail to match structure in a relevant document for a variety of reasons, thus structured queries, although containing more information than keyword queries, are often less effective than unstructured queries. This paper studies retrieval of sentences with annotations for a question answering task. Three problems of structured retrieval are identified and solutions proposed. Structural mismatch is addressed by query structure expansion of predicted relevant structures. Lack of presence of all key aspects of a question is solved by Boolean filtering of result sentences. The score variations of the annotator generated fields with all the different lengths are accounted for by using field specific smoothing. Experiments show that each solution incrementally improves structured retrieval, and a combination of Boolean filtering, structural expansion, and keyword queries outperforms keyword and simple structured retrieval baselines.

## Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: *Retrieval Models, Query formulation*

## General Terms

Algorithms, Experimentation, Theory

## Keywords

Structured Query Formulation, Boolean Filtering, Structural Mismatch, Indri Query Language, Question Answering

## 1. INTRODUCTION

Text retrieval using unstructured queries (*keyword retrieval*) assumes that a document consists of index terms (*keywords*), and a query consists of a (possibly weighted) set of index terms. Text retrieval for structured documents assumes that each index term is contained in a document component (historically called a *field*, more recently called an *element*), which may have been created by the author (e.g., HTML, XML) or an annotation process such as a sentence breaker, named-entity annotator, or semantic role labeler. A structured query can request documents or fields with complex field–field or field–term containment requirements. For example, the following query is a translation of the question "When did Wilt Chamberlain score 100 points" into the Indri query language [10], using ASSERT-style semantic role labels. It asks for sentences that have "score" as the target verb, a "date" named-entity field within the temporal argument (argm-tmp), "Wilt Chamberlain" within the agent field (arg0) of the target "score" and "100 points" to be in the patient field (arg1).

Q1: `#combine[sentence](`

```
#combine[target]( Score
    #combine[./argm-tmp]( #any:date )
    #combine[./arg0]( Wilt Chamberlain )
    #combine[./arg1]( 100 points )))
```

Each field restriction here should have a #max operator outside, e.g. `#combine[X](…)` is `#max(#combine[X](…))`, for any X. The #max operators are omitted in this paper for clarity sake.

Despite the fact that structured queries are strictly more expressive than keyword queries, there is no empirical work showing that automatically constructed structured queries, even if combined with keyword, can reliably outperform keyword queries. Prior work either relies on known correct relevant structures to construct queries [1] or extrapolates from known answer structure [2]. In XML retrieval, the INEX workshops [3] used manually created structured queries, but no significant improvement over keyword is observed [4]. Later work [2] showed that manually-corrected INEX queries could outperform keyword, but the improvement is only in top precision, over an NDCG measure. In TREC Legal tracks [6], carefully constructed Boolean queries (a type of structured query) have been shown to outperform keyword queries, but those queries are manually constructed through a careful and repeated negotiation process between adversarial experts that refines the Recall and Precision of a query, alternating until both sides are satisfied.

We focus on structured retrieval in support of a question answering (QA) application, with sentence retrieval as the task. We chose QA because semantic role labels of questions provide query structure. This paper identifies three problems that affect structured retrieval for question answering, and presents an automatic structured query formulation process that improves structured retrieval. Overall, the structured retrieval performance is brought to a level close to keyword. A significant improvement is gained by combining keyword and structured queries.

Our work is important for three reasons. First, it improves structured retrieval accuracy. Second, accurate structured queries are difficult to construct even manually, thus it provides guidelines for how to create accurate and robust structured queries. Finally, it shows that structure introduced by text annotations has important differences from more traditional forms of document structure (e.g., fields, XML structure).

In improving structured retrieval for texts with annotations, the contributions of this work include i) introducing Boolean filters that regulate structured queries by prohibiting false positives from being picked up by unstable structured queries, ii) a field-specific smoothing method that takes advantage of the fact that the optimal Dirichlet smoothing parameter depends on the average field length, and iii) modeling field level mismatches between query structure and the relevant answer structures, and using the model to predict possible relevant structures when given a new structured query. These contributions are not specific to question answering, but apply to structured retrieval in general.

## 2. BOOLEAN FILTERING

Best-match retrieval models typically calculate the score of a retrieved item by combining scores accrued by individual query terms. This effectively causes two problems when ranking results.

Firstly, most algorithms would give the term #any:date in query Q1 (above) a very low term weight (e.g. idf score), due to its corpus frequency. Matching a date has a smaller effect on an element's score than other query terms, thus increasing the


*CIKM'09*, November 2–6, 2009, Hong Kong, China.

likelihood of retrieving items that have *Wilt Chamberlain, score,* and *100 points*, but do not mention a date. However, the date is the expected answer type, and thus crucial for question answering success, as noted by Prager, et al. [7].

Secondly, when retrieving document elements such as sentences, the problem of false matches is exacerbated by retrieval models that smooth element scores using document-level information. Smoothing is generally considered desirable, because elements are so sparse [1, 2]. However, it also increases the likelihood of matching items that have *score*, *100 points*, a *date*, something in an `arg0` field, and *Wilt Chamberlain* elsewhere in the document.

These false positives appear because the simple structured queries do not express a Boolean constraint that is implicit but real in the topics and evaluation process: A relevant sentence must be self-sufficient; surrounding sentences are not taken to provide context. Keyword retrieval shares this problem, but it is more evident in structured retrieval.

Our solution is to make the Boolean constraint explicit. A Boolean filter is easily and automatically obtained by requiring at least one matching term for each field component of the query. For Q1, the filter is:

```
F1: #band(  #syn(Wilt Chamberlain)
            #syn(100 points)
            #any:date )
```

#band is the Boolean-AND operator in Indri query language and #syn is the Boolean-OR (synonym) operator.

The constraint F1 forms the set of sentences to be ranked. The query Q1 ranks the sentences that match the filter F1. In Indri's query language, this combination is expressed as a #filreq query:

```
#filreq (F1 Q1)
```

Note, the filter F1 does not require the target verb, because most of the times the verb will not match targets in relevant sentences.

This Boolean filtering strategy is general, and can be adapted for other retrieval tasks, as follows. *Firstly*, aspects of a topic are identified. In question answering, aspects of a question are defined as the argument fields of its semantic role parse. In ad-hoc retrieval with short keyword queries, each keyword might represent one aspect; for longer queries, terms might be grouped to form aspects. *Secondly*, expand each aspect by viable alternatives, and require all aspects to appear in a matching result. In QA, each term within an argument field (i.e. an aspect) is treated as a viable alternative to represent the aspect. For example, *Wilt Chamberlain* can be represented by either *Wilt* or *Chamberlain*. In ad hoc retrieval, synonyms and other alternative forms of a keyword aspect should be expanded. Search professionals such as librarians and lawyers have widely used these conjunctive normal form queries [6, 11, 12]. The use of these Boolean queries for filtering ranked results has also been shown effective [13, 14]. The contribution of this work is the automatic formulation of the Boolean filters using semantic role analyses of the questions.

To enforce the Boolean filter at the sentence level, do as below:

```
Q2: #combine[sentence]( #filreq( F1 Q1 ) )
```

On the AQUAINT corpus, this query returns only 71 sentences, but increases the Average Precision for keyword retrieval from 0.25 to 0.55, and Recall@1000 from 0.80 to 1.00. Mean AP for all test topics are presented in Section 6.2. For fair comparisons, structured query results were appended with keyword results to make the total number 1000 for each query.

Because of the extra matching, structured retrieval can be computationally expensive. Boolean filtering improves efficiency dramatically by reducing the number of items to be scored. These structured queries run at least as fast as keyword queries.

# 3. FIELD SPECIFIC SMOOTHING

Zhao and Callan pointed out that for structured retrieval, document and collection level (two-level) Dirichlet smoothing is more effective than two-level Jelinek-Mercer [2]. We follow the same two-level Dirichlet smoothing shown below.

$$P_s(q \mid F, D) = \frac{\mathrm{tf}(q,F) + \mu_d P_d(q \mid D, C)}{\mathrm{length}(F) + \mu_d} = \frac{\mathrm{tf}(q,F) + \mu_d \dfrac{\mathrm{tf}(q,D) + \mu_c P(q \mid C)}{\mathrm{length}(D) + \mu_c}}{\mathrm{length}(F) + \mu_d} \quad (2.1)$$

The above equation calculates the generation probability of the query "`#combine[`$F_t$`](` $q$ `)`" given a field $F$ in document $D$. $q$ is a query term, and $F_t$ is the type of the field $F$. $\mu_d$ is the document level smoothing parameter and $\mu_c$ the collection level smoothing parameter.

When applying the model to texts with annotations, $F_t$ can vary from short fields like targets, which have an average length 1, to sentences which average over 20 words. The optimal smoothing parameter depends on the average length of the resultant field being scored [2, 6]. Thus, our solution is to specify different $\mu_d$ and $\mu_c$ for each type of field, based on average field length. Among several methods, the best performing method for setting $\mu_d$ and $\mu_c$ for field f was as follows:

$\mu_d = L(f) * c_1$, $\mu_c = c(f)*L(f)*c_2$, where:

$c(f)$ is the average number of $f$ fields appearing in a document,

$L(f)$ is the average field length, and $c_1$, $c_2$ are tuned constants.

The same $c_1$ and $c_2$ are shared for all fields, thus field-specific smoothing is obtained using only two corpus-level constants and two field-specific statistics. Implementing this solution in Indri is easy. Indri already supports the use of different smoothing rules for different fields. We just create a rule for each type of field.

# 4. FIELD MISMATCH MODELING

Structured queries usually specify what fields are expected, and what terms should appear in each field. Mismatch in the structure is a frequent problem. Automatic annotators make mistakes, causing structure level mismatch between the query and relevant texts. Natural language usage is versatile, leading to even more mismatches.

For example, for Q1, a relevant but low ranked sentence can be, S1: "*In 1962 when he scored 100 points in a single game, Wilt Chamberlain lived in an apartment in ….*" In sentence S1, there are two target verbs, *score* and *live*, and *Wilt Chamberlain* is not the agent (arg0) of *score*. When scoring sentence S1 with query Q1, the final (maximal) score will come from the answer structure with *score* as the `target`, *100 points* as `arg1`, and `arg0` (*Wilt Chamberlain*) being scored through smoothing because of no match to any of the arguments of the target *score*. If we assume that whenever a field is unmatched, it is aligned to the sentence for back-off, then the maximal alignment from fields in Q1 to fields of sentence S1 would be `target` to `target` (score), `arg1` to `arg1` (100 points), `argm-tmp` to `argm-tmp` (in 1962), but `arg0` to the sentence.

We propose to make use of imperfect annotations by modeling the structural mismatch between query and document and to still produce better retrieval.

## 4.1 Question-Answer alignment

To show field level mismatches between questions and answers (Q-A), one can follow the above alignment process, matching questions to the known relevant answers. Field level alignment statistics are shown in Table 1.

The table gives counts of alignments normalized by the number of topics/queries. For example, target verb matches itself in only 14% of the cases, and it is even worse with other argument fields.

## 4.2 Modeling field mismatches

We want to estimate given a structured query, what are the likely alternative structures and how likely they are. However, because we have a limited number of topics, about 50 for training, there are not enough data to build a model that conditions on query terms. We condition the model on the field types being queried.

The baseline model *Ind* treats all arguments independently. An argument in the query can be matched to answer fields according to a multinomial distribution over all possible argument types.

**Table 1.** Excerpt confusion matrix for argument fields in the queries v.s. aligned arguments in the relevant answers. Each row shows how many times that type of field in the query is aligned to fields in the relevant sentences, divided by the total number of topics/queries. Listed fields are only an excerpt of the 13 fields that appear in more than 20% of the documents (this set contains all the queried fields in the topics). Highest value in row is bolded, if it is the sentence field, next highest value is bolded.

| Q\A | arg0 | arg1 | arg2 | tmp | sentence | target |
|---|---|---|---|---|---|---|
| **arg0** | **.1000** | .0917 | .0150 | .0017 | 0.3167 | 0 |
| **arg1** | .0583 | **.2917** | .0483 | .0550 | 0.4683 | 0 |
| **arg2** | .0117 | **.0450** | .0150 | .0250 | 0.2017 | 0 |
| **tmp** | .0133 | .0417 | .0067 | **.1117** | 0.2033 | 0 |
| **target** | 0 | 0 | 0 | 0 | 0.8600 | **0.1400** |

The parameters for this model – P($A$ | $B$) for any field $A$, $B$ – are just the row-wise normalized version of Table 1. Given a test query with $n$ fields ($X_1$, $X_2$, .. $X_n$), an alternative structure ($Y_1$, $Y_2$, .. $Y_n$) is scored by multiplying the translation probabilities together P($Y_1|X_1$)*P($Y_2|X_2$)*..* P($Y_n|X_n$).

We also propose the Cooc model, a Markov network model, which captures co-occurrences of argument alignments, when assigning probabilities to alternative answer structures.

We use a Markov network model as illustrated in Figure 1. We model one particular field alignment $A$:$B$ as a binary random variable, where $A$ is the queried field and $B$ is the aligned answer field. $A$:$B$ equals 1 only if the queried field $A$ is aligned to field $B$ in the answer structure. This way we can model co-occurrence of alignments, instead of the baseline independent model. For example, arg0:arg0 and arg1:arg1 would co-occur in a perfectly matched Q-A pair with the query and the answer both having arg0 and arg1 fields. To encourage alignments similar to training data, edge weights between the alignment nodes $W$($A$:$B$, $C$:$D$) are defined as co-occurrence counts Co-occur($A$:$B$, $C$:$D$). No edges are added between alignment variables that did not co-occur. The edge potential $\psi$($A$:$B$, $C$:$D$) is further defined as follows:

| $\psi$($A$:$B$, $C$:$D$) | $C$:$D$ = 0 | $C$:$D$ = 1 |
|---|---|---|
| $A$:$B$ = 0 | $\beta_1$ $W$($A$:$B$, $C$:$D$) | $\beta_3$ |
| $A$:$B$ = 1 | $\beta_2$ | $\beta_4$ $W$($A$:$B$, $C$:$D$) |

$\beta_{1...4}$ are the tuning parameters shared by all edges. If the learning algorithm is reasonable, typically $\beta_2$ and $\beta_3$ will be small, penalizing co-occurrences of alignments inconsistent with the training data, and $\beta_4$ will be large, encouraging observed co-occurrences to appear in the predictions.

When an assignment to all graph nodes is given, we know which alignment variables equal 1. By collecting these activated alignments, we get a whole alignment for the query and thus the predicted answer structure (see examples in Section 5).

During testing, all query field variables are observed, while all alignment variables are unknown. For a particular assignment to all the alignment variables, we calculate the total graph potential, and rank it using the potential.

Iterative proportional fitting (IPF) with loopy belief propagation can be used to train the model. However, for this graph, IPF does not converge. Since the small number of parameters to train, we did grid search over the parameter space using prediction accuracy as the measure to select the optimal parameters.

The complexity of calculating the potential is linear in the size of the graph (edge + nodes), and the complexity of enumerating all possible structures for a given query is exponential in the size of the structure. For this task, the largest queries only have a small number of fields, and we calculate exact potentials and rankings for all variant structures of a given test query.

In experiments, the Ind and Cooc models performed similarly.

# 5. QUERY FORMULATION

The foundation of the structured queries is the structure assigned by the semantic role labeler. The raw structured queries are just like Q1 (Section 1). Adding the Boolean filter changes the query from Q1 to Q2 (Section 2).

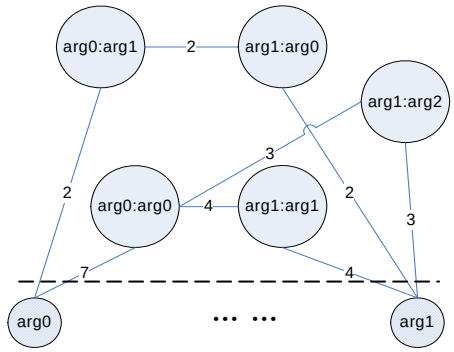


**Figure 1.** An illustration of the Cooc model

For the structure prediction models, the query formulation is more interesting. The structure prediction models produce sets of alternative structures for a given query. Take Q1 for example, if the top two predicted alignments are "arg0:sentence arg1:arg1 argm-tmp:argm-tmp target:sentence" and "arg0:sentence arg1:sentence argm-tmp:sentence target:sentence", the two predicted structured queries will be

```
Q1.1: #combine[sentence](
        score wilt chamberlain
        #combine[target](
           #combine[./arg1]( 100 points )
           #combine[./argm-tmp](#any:date) ) )
Q1.2: #combine[sentence](
      score wilt chamberlain 100 points #any:date)
```

From this prediction, we simply take the maximum score of the two predictions to produce the final score. Thus, the resulting query will be

```
Q3: #combine[sentence]( #filreq(
          F1 #max( Q1 Q1.1 Q1.2 )))
```

Q3 uses the predicted structures as back-offs for the original structure. The number of alternative structures to use, $\alpha$, is a tuning parameter that will be trained.

This structured query is different from the keyword query (call it KW, which happens to be Q1.2 above), while its performance is close to the keyword query. Thus, we further combine it with KW, which leads us to the final query:

```
Q4: #combine[sentence]( #filreq ( F1
      #weight( w_o KW w_s #max( Q1 Q1.1 Q1.2 ))))
```

Here $w_o$ and $w_s$ are weights. $w_o$ is the weight on the keyword query, and $w_s$ is just (1 – $w_o$). We put KW inside the filter so that the filter regulates it as well as the structured query.

# 6. EXPERIMENTS

## 6.1 Datasets

We used two datasets. The first is the AQUAINT corpus with TREC 2002 factoid QA topics. The documents are segmented into 21 million sentences, and sentence level relevance judgments are the same as [1, 2]. A total of 109 topics were split randomly into training and test sets of 55 and 54 topics respectively.

We also included a smaller dataset, containing the development and test sets created by Wang et al [9]. 133 factoid QA topics from TREC 2004 were split into 65 training and 68 test topics. Sentence breaking and sentence level judgements were provided based on answer string patterns and document level judgements provided by TREC. Because it was intended to test sentence reranking, only top rank sentences with fewer than 40 words are included. We include this set despite characteristic differences.

Questions were first transformed using simple syntactic rules into statements using the OpenEphyra system version 0.1.1 [10], and then parsed by ASSERT version 0.14b. Answer types were also predicted by OpenEphyra. ASSERT does not parse for some verbs such as “be”, “have” and “become”, for the TREC 2002 set, questions were paraphrased manually into verbs that ASSERT

**Table 2.** Sentence retrieval evaluated in MAP on the TREC 2002 test set. SRL is the baseline semantic role label structured queries, and each method in the next column is added to the previous method, the change from adding each method is shown in the 3rd row. Running time of each method is shown in the last (4th) row.

| SRL | +Filter | +Smooth | +Cooc | Keyword | Mixed |
|---|---|---|---|---|---|
| 0.0675 | 0.1236 | 0.1426 | 0.1707 | 0.1815 | 0.2105 |
| 0% | +83.1%* | +15.4%† | +19.7% | 0% | +16.0%‡ |
| 35.7m | 3.97min | 3.97min | 9.23m | 18.6min | 9.5min |

* Significant by paired sign test at $p < 0.0001$, degree freedom = 29.
† Significant by sign test at $p < 0.05$, degree of freedom = 27.
‡ Significant by sign test at $p < 0.009$, degree of freedom = 34.

**Table 3.** Sentence retrieval evaluated in MAP on the TREC 2004 test set. Same layout as in Table 2 is used, except that running time is dropped because the test set is too small to observe a difference.

| SRL | +Filter | +Smooth | +Cooc | Keyword | Mixed |
|---|---|---|---|---|---|
| 0.4360 | 0.4341 | 0.4513 | 0.4193 | 0.3153 | 0.4193 |
| 0% | -0.416% | +3.95% | -7.08% | 0% | +33.0%* |

* Significant by paired sign test at $p < 0.022$, degree freedom = 10.

does parse. For the TREC 2004 dataset, these questions were discarded, leaving 22 topics for training and 22 topics for testing.

## 6.2 Sentence retrieval

To see how much of improvement each of the discussed methods brings, we present, in Table 2, the overall sentence retrieval performance on the TREC 2002 test set. In this experiment, the semantic role structure was manually labeled by someone else, and is more accurate than automatically assigned by OpenEphyra + ASSERT.

The Boolean filtering and per field smoothing both significantly increased retrieval accuracies. Adding the Cooc model increased performance by 19.7%, though not significant by sign test. The same trends are observed on the training set. By merging the keyword query into its structured correspondence as Q4 of Section 5, the performance of the structured queries became stable and consistently outperformed keyword.

Table 2 shows the potential of the retrieval strategies with the gold standard semantic analyses. We also did experiments using automatic semantic analysis and answer type analysis for the TREC 2002 questions. 35 test topics had semantic role output by OpenEphyra + ASSERT, and were used as input for the retrieval strategies. +Filter and Mixed were still significantly better than their baselines.

In Table 3, we also list results for the TREC 2004 test set. The SRL structures were assigned automatically by OpenEphyra + ASSERT. Here, the SRL baseline was already high above keyword retrieval, though not statistically significant. This was largely due to a smaller collection with fewer false matches for the structures. The +Cooc model was the best Mixed model and it was significantly better than keyword. This shows the Cooc model captures some of the mismatches and also has the effect of backing-off to keyword retrieval. Most of the differences shown in Table 3 are small and insignificant due to a small set of topics. However, the mixed approach was still robustly outperforming keyword retrieval. Across both datasets, this mixed approach is robust and consistently outperforming the baseline.

# 7. CONCLUSIONS

We identified issues of structured retrieval using text annotations, improved structured queries significantly, and showed significant improvements over their keyword counterpart through merging the structured with the keyword queries. This advancement is achieved at a relatively low computational cost.

*Firstly*, there are Boolean constraints implicit in the topics that were not enforced by score-based retrieval models. This causes more problems in structured retrieval using text annotations, because of the (fake) word matches added to short fields by smoothing. An automatic and simple way of formulating keyword Boolean filters was designed. The Boolean filtering enforces the presence of necessary keywords, and significantly boosts retrieval for both keyword and structured queries. Filtering also reduces processing time.

*Secondly*, text annotations produce field types of very different lengths, and thus require different amounts of smoothing. A field specific Dirichlet smoothing method is introduced and shown to improve structured retrieval. Per field smoothing allows different smoothing parameters to be applied to different fields. Parameters are proportional to the average field lengths for easy tuning.

*Thirdly*, to handle the structural mismatch between query and text, maximal alignments of Q-A pairs were used to show the severity of the problem and provide training data. A Markov random field model is used to globally enforce the constraint that predicted alignments should be as consistent with training observations as possible. It showed reasonable performance and boosted the performance of the structured queries further.

*Lastly*, the best performance is achieved by mixing the boosted structured queries with the keyword queries, using the advantages of both approaches. This accuracy gain is achieved, at little extra costs, efficiency-wise as compared to keyword retrieval.

The work reported here improves retrieval accuracy, but more importantly, it provides greater understanding of the problems that affect structured retrieval, especially when applied to structure introduced automatically by text annotation processes. This work also provides guidance for creating high quality structured queries.

# 8. ACKNOWLEDGEMENTS

We thank Matthew Bilotti and reviewers for comments on the work. This work is supported by National Science Foundation grant IIS-0707801 and IIS-0534345. The views and conclusions are the authors', and do not reflect those of the sponsor.

# 9. REFERENCES

[1] Matthew W. Bilotti, Paul Ogilvie, Jamie Callan and Eric Nyberg. Structured Retrieval for Question Answering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 2007.

[2] Le Zhao and Jamie Callan. A Generative Retrieval Model for Structured Documents. In *Proceedings of CIKM* 2008.

[3] Norbert Gövert and Gabriella Kazai. Overview of the INitiative for the Evaluation of XML retrieval (INEX) 2002, 2002.

[4] Andrew Trotman and Mounia Lalmas. Why Structural Hints in Queries do not Help XML-Retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp711-712. 2006.

[5] Hui Fang, Tao Tao and Chengxiang Zhai. A formal study of information retrieval heuristics. In *Proceedings of the 27th international ACM SIGIR conference on Research and development in information retrieval*. 2004.

[6] TREC Legal track homepage (retrieved on Jan 2. 2009): http://trec-legal.umiacs.umd.edu/

[7] John Prager, Jennifer Chu-Carroll, Eric Brown, Krzysztof Czuba. Question Answering By Predictive Annotation. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. 2000.

[8] INDRI - Language modeling meets inference networks. http://www.lemurproject.org/indri/. Retrieved Jan 2, 2009.

[9] Mengqiu Wang, Noah Smith and Teruko Mitamura. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. *Proceedings of EMNLP '07*.

[10] Nico Schlaefer, Jeongwoo Ko, Justin Betteridge, Guido Sautter, Manas Pathak and Eric Nyberg. Semantic Extensions of the Ephyra QA System for TREC 2007. In *Proceedings of the Sixteenth Text REtrieval Conference*, TREC 2007.

[11] Wilfrid Lancaster. *Information Retrieval Systems: Characteristics, Testing and Evaluation*. Wiley, New York, 1968.

[12] Stephen Harter. *Online information retrieval: concepts, principles, and techniques* Academic Press, San Diego, California, 1986.

[13] Marti Hearst. Improving full-text precision on short queries using simple constraints. In *Proceedings the Fifth Annual Symposium on Document Analysis and Information Retrieval (SDAIR 1996)*, 1996.

[14] Mandar Mitra , Amit Singhal and Chris Buckley, Improving automatic query expansion, *Proceedings the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, p.206-214, 1998.