Rank Learning for Factoid Question Answering with Linguistic and Semantic Constraints

Matthew W. Bilotti, Jonathan Elsas, Jaime Carbonell and Eric Nyberg
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213, USA
{ mbilotti, jelsas, jgc, ehn }@cs.cmu.edu

ABSTRACT

This work presents a general rank-learning framework for passage ranking within Question Answering (QA) systems using linguistic and semantic features. The framework enables query-time checking of complex linguistic and semantic constraints over keywords. Constraints are composed of a mixture of keyword and named entity features, as well as features derived from semantic role labeling. The framework supports the checking of constraints of arbitrary length relating any number of keywords. We show that a trained ranking model using this rich feature set achieves greater than a 20% improvement in Mean Average Precision over baseline keyword retrieval models. We also show that constraints based on semantic role labeling features are particularly effective for passage retrieval; when they can be leveraged, an 40% improvement in MAP over the baseline can be realized.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Question answering, passage retrieval, learning to rank, annotation graphs, text annotations, committee perceptron

1. INTRODUCTION

Question Answering (QA) systems aim to deliver specific answers to user questions posed in natural human language. A QA system can be thought of as an embedded passage retrieval process bookended by Natural Language Processing (NLP) components that allow the system, to understand the question, on the front end, and post-retrieval, to locate answers among the results. If QA systems are ever to become competitive with the *ad hoc* keyword search engines that are ubiquitous in the lives of today's internet users, both latency and accuracy must be improved. Both of these goals can be addressed by improving the quality of the embedded passage retrieval component.

Poor passage retrieval quality within QA systems stems in part from a mismatch between what the system wants and what the embedded retrieval component is able to query. Internally, QA systems represent their *information needs* as sets of linguistic and semantic constraints that a retrieved passage must satisfy if it answers the question. Many passage retrieval approaches commonly used in QA systems can not check these types of constraints at query time. As a result, QA systems are forced to approximate their information needs in terms of classic *ad hoc* retrieval primitives such as bag-of-words, proximity and named entity features.

For many questions, the classic feature set poorly approximates the information need, resulting in the retrieval of too few answer-bearing passages and/or too many false positives. This degradation in passage retrieval quality overburdens the downstream Answer Generation component, which must determine whether each retrieved passage is answer-bearing by comparing it against the linguistic and semantic constraints specified in the information need. Post-retrieval constraint-checking involves potentially slow NLP analysis, which can limit the number of results that can be considered in the time available. In addition to increasing system latency, poor quality retrieval results can also degrade accuracy when the best answer is ranked so low that the system does not have a chance to consider it. The focus of this paper is on improving the quality of results ranked by a QA system's embedded passage retrieval component, thereby providing the best possible foundation for a fast and accurate end-to-end system.

In this paper, we propose a new approach to passage retrieval for QA systems, based on rank-learning techniques, that integrates linguistic and semantic constraint-checking into the retrieval process. The approach utilizes a novel method of decomposing the question representation, viewed as a graph, into atomic constraints to be matched in candidate answer-bearing passages. This decomposition enables partial constraint matching, differential weighting of constraint types, and graceful back-off to baseline ranking features such as bag-of-words and named entity matches. Atomic constraints become features for a trained model able

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.
Copyright 2010 ACM 978-1-4503-0099-5/10/10 ...$10.00.

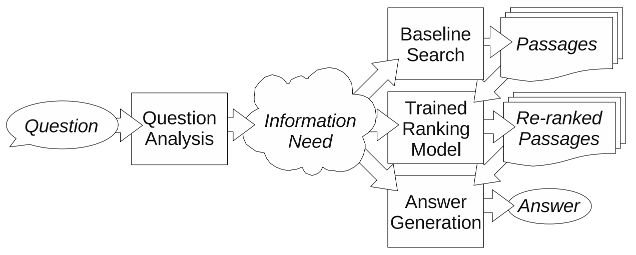

Figure 1: Block diagram of a pipelined QA system. This paper studies the impact on passage retrieval quality of re-ranking baseline search results using a trained model capable of checking the linguistic and semantic constraints specified in the system's information need.

to judge linguistic and semantic similarity between a retrieved passage and the QA system's information need. We show that this model provides significant improvements in Mean Average Precision when used to re-rank passages retrieved by a baseline retrieval approach consisting of bag-of-words and named entity features.

2. PASSAGE RETRIEVAL FOR QA

Consider a Question Answering (QA) system with broad coverage on the common types of factual, short-answer questions known as *factoids* within the research community. For the purposes of this study, we consider only the first two modules of the pipelined QA system shown in Figure 1. The Question Analysis module uses NLP resources to map the question into a representation we call the *information need*, which consists of the linguistic and semantic constraints that an answer-bearing passage must satisfy. The information need serves as input to a Baseline Search module responsible for retrieving text likely to be relevant to that information need.

The experiments in this paper will focus on measuring the improvement in passage retrieval quality gained by introducing our linguistic and semantic rank-learning approach, depicted as the Trained Ranking Model in Figure 1. The model re-ranks the top 1000 baseline results with respect to the linguistic and semantic constraints expressed in the information need. The evaluation compares the quality of the trained model's passage ranking with respect to the baseline in terms of Mean Average Precision. The Answer Generation module shown in the figure is included for illustration purposes only, as end-to-end system accuracy is not explicitly evaluated in this paper[1].

The baseline passage retrieval approach in our QA system consists of bag-of-words Indri[2] queries with named entity placeholders to represent the expected answer type. When scoring passages, as opposed to entire documents, this approach approximates density-based methods considered to be strong baselines for QA [22]. For example, consider question 1398 from the QA track at TREC[3] 2002, *What year was Alaska purchased?* The answer is *1867*. The baseline Indri query combines question keywords with an `#any:` operator matching occurrences of `date`, the expected answer type. The extent restriction operator, `[sentence]`, retrieves and scores sentence-sized passages individually.

```
#combine[sentence]( #any:date year Alaska purchased )
```

[1]Elsewhere, it is shown that an improved passage ranking in terms of MAP can translate to improved end-to-end system accuracy or answer Mean Reciprocal Rank (MRR) [2].

[2]See: `http://www.lemurproject.org/indri`

[3]Text REtrieval Conference. See: `http://trec.nist.gov`

$$G = (E = \{e_1, ..., e_{|E|}\}, R = \{r_1, ..., r_{|R|}\}, T)$$
$$T = (Te = \{te_1, ..., te_{|Te|}\}, Tr = \{tr_1, ..., tr_{|Tr|}\})$$
$$te_i = (name, parent \vee \emptyset)$$
$$tr_i = (name, te_d, te_r); te_d, te_r \in Te$$
$$e_i = (te); te \in Te$$
$$r_i = (tr, e_d, e_r); tr \in Tr; e_d, e_r \in E$$

Figure 2: Formal description of an annotation graph G, consisting of a set of elements E, a set of relations R, and a type system T, which defines element types Te and relation types Tr.

3. REPRESENTING THE INFORMATION NEED AS AN ANNOTATION GRAPH

We assume that a QA system analyzes its input question into an information need representation containing linguistic and semantic constraints that can be used to rank passages. Furthermore, we make the weak assumption that the system's information need can be represented as an *annotation graph*, a generalized formalism described in this section. Annotation graphs make it easy to represent keywords and arbitrary linguistic and semantic relationships between them.

Figure 2 gives a formal description of an annotation graph G, consisting of a set of elements E, a set of relations R, and a type system T. A type system defines sets of element and relation types, notated Te and Tr, respectively. Each element type te_i has a name and a pointer to an optional parent element type. Each tr_i is a named relation type defined to hold over specific domain and range element types, or types that inherit from them. The inheritance mechanism for element types allows for relations defined over element supertypes to be instantiated over instances of subtypes of those elements.

The QA system in this paper represents its information need under the type system T_{ne+srl}, defined in Figure 3. It supports common named *entity* types and semantic role labeling, in which `target` verbs are related to their *arguments*, which are assigned PropBank [14] semantic roles by the ASSERT semantic parser [17]. The type system models *enclosure* relations between *field* pairs, and between *fields* and *keywords*, *ordering* relations between *keyword* pairs, and *attachment* relations between `targets` and their *arguments*.

Figure 4 shows the annotation graph representing the information need for our example question. Question *keywords year*, *Alaska* and *purchased* are in the bottom row, and the pairwise *ordering* relations holding among them are shown with curved arrows. *Alaska* was recognized as a `location`, so there is an *enclosure* relation between them shown as a solid

$$T_{ne+srl} = (Te, Tr)$$

$$Te = \left\{ \begin{array}{ll} (\mathit{field}, \emptyset), & (\texttt{arg0}, \mathit{argument}), \\ (\mathit{keyword}, \emptyset), & (\texttt{arg1}, \mathit{argument}), \\ (k \in V, \mathit{keyword}), & (\texttt{arg2}, \mathit{argument}), \\ (\texttt{sentence}, \mathit{field}), & (\texttt{arg3}, \mathit{argument}), \\ (\texttt{target}, \mathit{field}), & (\texttt{arg4}, \mathit{argument}), \\ (\mathit{argument}, \mathit{field}), & (\texttt{argm-adv}, \mathit{argument}), \\ (\mathit{entity}, \mathit{field}), & (\texttt{argm-dir}, \mathit{argument}), \\ (\texttt{date}, \mathit{entity}), & (\texttt{argm-loc}, \mathit{argument}), \\ (\texttt{location}, \mathit{entity}), & (\texttt{argm-mnr}, \mathit{argument}), \\ (\texttt{org}, \mathit{entity}), & (\texttt{argm-tmp}, \mathit{argument}) \\ (\texttt{person}, \mathit{entity}), & \end{array} \right\}$$

$$Tr = \left\{ \begin{array}{l} (\mathit{enclosure}, \texttt{sentence}, \mathit{field}), \\ (\mathit{enclosure}, \mathit{argument}, \texttt{target}), \\ (\mathit{enclosure}, \mathit{argument}, \mathit{argument}), \\ (\mathit{enclosure}, \mathit{argument}, \mathit{entity}), \\ (\mathit{enclosure}, \mathit{field}, \mathit{keyword}), \\ (\mathit{ordering}, \mathit{keyword}, \mathit{keyword}), \\ (\mathit{attachment}, \texttt{target}, \mathit{argument}) \end{array} \right\}$$

Figure 3: Definition of the type system used in this paper, which supports named entities and ASSERT semantic role labeling. Note that there is an element type for each keyword k in the vocabulary V. Leaf element types representing text annotations are given in `monospace type`.

arrow. The `sentence` participates in an *enclosure* relation with all of the other *fields* and *keywords* in the graph.

The semantic structure in the question is modeled by the `target`, `argm-tmp` and `arg1` *arguments*, the *enclosure* relations between them and the *keywords*, and the *attachment* relations between the `target` and the *arguments*, shown as dashed arrows. In this question, *Alaska* is labeled as `arg1` because it is the thing being *purchased*; the buyer (`arg0`) is not specified. The answer is expected to be of type `date` and to occur in the temporal adjunct labeled `argm-tmp`.

4. LEARNING TO RANK FOR FACTOID QA

This section describes our approach to learning to rank passages according to their linguistic and semantic similarity to an information need. In Section 4.1, we describe how to select, for any given type system, a set of atomic linguistic and semantic constraints, which are represented as annotation graph snippets. We show an example of this decomposition for T_{ne+srl}, the type system underlying the experiments in this paper, in Section 4.2. In Section 4.3, we describe how atomic constraints become features useful for ranking and introduce the learning algorithm used for the experiments in this paper.

4.1 Selecting Atomic Constraints by Type System Decomposition

This section presents an algorithm for the decomposition of a type system into atomic constraints based on the element and relation types defined in the type system. All constraints are annotation graphs, and the *order* of a constraint

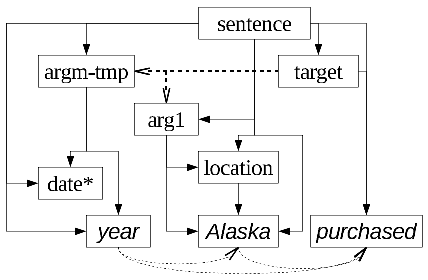

Figure 4: Information need representation for the question, *What year was Alaska purchased?* Solid arrows indicate *enclosure* relations, dashed arrows indicate `target`-*argument attachment* relations, and curved arrows indicate *keyword ordering* relations. The asterisk indicates the expected answer type.

Table 1: Algorithm for Type System Decomposition

```
Input:
Type system $T = (Te, Tr)$
Maximum order $n$ (to generate constraints of order $1 \ldots n$)
Output:
Set of sets of constraints, one for each order $\{C_1, \ldots, C_n\}$

  1. Initialize $C_{1 \ldots n} = \emptyset$
  2. For each $tr = (name, domain, range) \in Tr$:
         For each $d, r \in Te$ such that $d$ and $r$ are concrete leaf
         types inheriting from $domain$ and $range$, respectively:
             Add to $C_1$: $\left( \begin{array}{l} E = \{(e_1, d), (e_2, r)\}, \\ R = \{(name, e_1, e_2)\}, T \end{array} \right)$
  3. For $i = 2, \ldots, n$:
         For each $tr = (name, domain, range) \in Tr$:
             For each constraint $c = (E_c, R_c, T) \in C_{i-1}$:
                 For each element $e \in E_c$ of type $domain$ or a
                 subtype:
                     For each $r \in Te$ such that $r$ is a concrete
                     leaf type inheriting from $range$:
                     Add to $C_i$:
                     $\left( \begin{array}{l} E = E_c \cup \{(x, r)\}, \\ R = R_c \cup \{(name, d, x)\}, T \end{array} \right)$
                 For each element $r \in E_c$ of type $range$ or a
                 subtype:
                     For each $d \in Te$ such that $d$ is a concrete
                     leaf type inheriting from $domain$:
                     Add to $C_i$:
                     $\left( \begin{array}{l} E = E_c \cup \{(x, d)\}, \\ R = R_c \cup \{(name, x, r)\}, T \end{array} \right)$
Note: $keyword$ is always considered a leaf type for the purposes
of this algorithm.
```

is equal to the number of relations in the graph. A first-order constraint, therefore, consists of two elements joined by a single relation.

The algorithm shown in Table 1 enumerates in set C_1 all of the possibilities for first-order constraints by reading the type system's set of relation definitions Tr and inserting all possible domain and range element types allowable. The main loop of the algorithm begins at step 3. At each iteration, the algorithm builds a set of order-i constraints based on the constraints of order $i-1$. The algorithm builds higher-order constraints by extending lower-order constraints, adding a single element and relation each time. The algorithm maintains a list of constraints as it traverses the type system in a breadth-first fashion.

To illustrate this process by example, consider the following first-order constraint, which contains a single *enclosure* relation between an `arg1` and a `person`:

$$\left(\begin{array}{l} E = \{(a_1, \texttt{arg1}), (p, \texttt{person})\}, \\ R = \{(\mathit{enclosure}, a_1, p)\}, T_{ne+srl} \end{array} \right)$$

To build a second-order constraint, the algorithm can introduce an *attachment* relation with a `target` as its source:

$$\left(\begin{array}{l} E = \{(t, \texttt{target}), (a_1, \texttt{arg1}), (p, \texttt{person})\}, \\ R = \{(\mathit{attachment}, t, a_1), (\mathit{enclosure}, a_1, p)\}, T_{ne+srl} \end{array} \right)$$

The worst-case complexity of this algorithm occurs for a type system for which every relation can be defined over any pair of element types, in either direction. The number of first-order constraints would be:

$$2 \cdot \left(\begin{array}{c} |Te| \\ 2 \end{array} \right) \cdot |Tr|$$

For each constraint of order $i-1$, the algorithm constructs an order-i constraint by adding a new element and a new relation. In the worst case type system, any element could be chosen to be extended by any new relation in any direction and with any new element at the other end of the relation. The number of new constraints added would be:

$$2 \cdot i \cdot |Te| \cdot |Tr|$$

The overall complexity of the algorithm is:

$$2 \cdot \left(\begin{array}{c} |Te| \\ 2 \end{array} \right) \cdot |Tr| \cdot \prod_{i=2}^{n} 2 \cdot i \cdot |Te| \cdot |Tr|$$

$$= 2^n \cdot n! \cdot \left(\begin{array}{c} |Te| \\ 2 \end{array} \right) \cdot |Te|^{n-1} \cdot |Tr|^n$$

In the next section, we show the decomposition for T_{ne+srl}, which does not approach the worst case complexity. It is likely that more efficient algorithms exist for type system decomposition, but we leave their design and implementation for future work.

4.2 Linguistic and Semantic Constraints for T_{ne+srl}

This section describes the linguistic and semantic constraint types obtained by decomposing T_{ne+srl}, the type system used in the experiments in this study. Though this feature set provides a comprehensive basis for annotation graph similarity, we opt to reduce computational complexity in practice. To limit the number of constraints selected by the algorithm, we set the maximum order $n = 3$. In an effort to avoid redundant or sparsely predictive features, we prune the constraint sets C_i for $i > 2$ according to the following set of principles derived from intuition[4] about the NLP tools underlying T_{ne+srl}:

1. A *field* may participate in at most one *enclosure* relation with another *field*. The underlying NLP tools enforce transitivity of enclosure, so the first-order constraints with a single *enclosure* relation are sufficient to capture all of the information.

2. When *attachment* relations are present between `targets` and `arguments`, consider only *keyword enclosure*. The *attachment* and *enclosure* relations can be thought of as independent, so they are matched in separate constraints.

3. Do not model *enclosure* of `targets` and *arguments* within other *arguments*. Though ASSERT allows this, the interesting linguistic and semantic relations can be captured by treating nested `target`-*argument* structures as siblings.

4. For *keyword ordering*, *keywords* must be enclosed in the same *field*. If the *keywords* were in separate *fields*, or none at all, the constraint would never be satisfied.

After running the decomposition algorithm and thinning the number of constraints using the above-described principles, we are left with eight classes of atomic linguistic and semantic constraints, which are shown in Figure 5, and are discussed individually below, with numbers referring to the figure. One additional constraint class mentioned below, **Paths(** N **)**, is not generated using the algorithm, but using a separate procedure described below.

1. **KEnc(** *field* **)**: This constraint is satisfied by a passage containing an *enclosure* relation between an instance of the given *field* and a specific *keyword* that is also enclosed by that *field* in the information need.
2. **KOrd(** *field* **)**: A passage satisfies this constraint if it contains a pair of *keywords*, an instance of the given *field*, an *enclosure* relation between the *field* and each of the *keywords* and a *ordering* relation between the *keywords*, as specified in the information need.
3. **FEnc(** *field*$_1$**,** *field*$_2$ **)**: A passage with an *enclosure* relation between an instance of $field_1$ and an instance of $field_2$ satisfies this constraint.
4. **Att(** *argument* **)**: This constraint checks for an *attachment* relation between a `target` instance and the given *argument*.
5. **Ans**: This constraint is equivalent to "FEnc(`sentence`, *entity*$_1$)" where *entity*$_1$ is equal to the *entity* type expected to be the answer to the question, as specified in the information need. If it is not specified, any passage can satisfy this constraint.
6. **Args(** N **)**: A passage satisfies this constraint if it contains N *attachment* relations between a single `target` instance and N *argument* types, as specified in the information need.

[4]More principled methods of feature selection, such as that proposed by Geng, et. al. [10], could be applied here, but we leave that for future work.

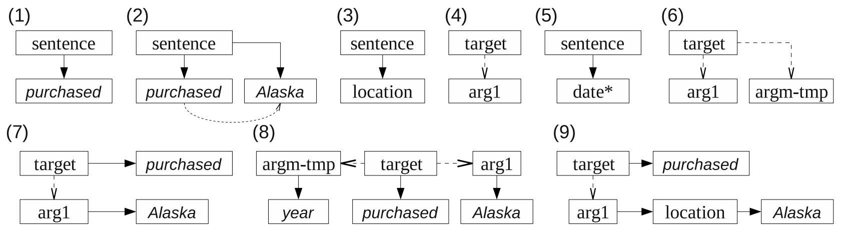

Figure 5: Annotation graph snippets representing linguistic and semantic constraints.

7. **Ta(** *argument* **)**: To satisfy this constraint, a passage must match two *keywords* from the information need, as well as the following relations: *enclosure* between a `target` instance and one of the *keywords*, *enclosure* between the given *argument* type and the other *keyword*, and *attachment* between the `target` and the *argument*.
8. **Taa(** $argument_1$**,** $argument_2$ **)**: This constraint is similar to "Ta(*argument*)", but checks for an instance of each of the given *argument* types each having an *attachment* relation with the same `target`. To satisfy this constraint, a passage must match three *keywords* from the information need, one each having an *enclosure* relation with the `target`, $argument_1$ and $argument_2$.
9. **Paths(** N **)**: This powerful constraint requires a passage to match a path through the annotation graph between two *keywords* specified in the information need. The path is not determined statically, at the time of type system decomposition, but rather dynamically; any path between two *keywords* that exists in the information need can match against the passage. The path consists of *N enclosure* and *attachment* relations between *argument*, *entity* and `target` elements in the graph. A path of length zero means that the two *keywords* are enclosed in the same *field*.

4.3 Rank Learning with Constraint Features

The above-described constraints are snippets of annotation graphs that can be compared against passages represented as annotation graphs to determine whether the passages *satisfy* the constraints. A constraint is considered satisfied by a passage if there exists a sub-graph alignment of the constraint annotation graph to the passage annotation graph[5]. A valid alignment is a mapping from elements of the constraint sub-graph to those of the passage graph such that all mapped elements are of the same type, and all relations that hold between elements in the constraint sub-graph also hold between the mapped elements in the passage graph. Satisfied constraints become features useful for ranking by counting the number of distinct alignments of the constraint to the passage that exist. See Figure 6 for an illustration of a constraint graph aligning to a passage annotation graph.

For T_{ne+srl}, we generate 162 count-based features by counting the number of distinct sub-graph alignments to a passage annotation graph for each constraint annotation graph. Leveraging this large number of features requires carefully

[5]Sub-graph alignment is known to be an NP-complete problem, but the problem instances are small enough to be tractable for this application.

Table 2: Full Feature Set, with group membership. Features are parameterized by element types defined in Figure 3. N is the path length through the annotation graph.

Feature Name	Groups
Baseline retrieval score	1-8
KEnc(`sentence`)	1,2,6,7,8
KOrd(`sentence`)	2,4,7,8
KEnc(*entity*)	3,4,8
FEnc(`sentence`, *entity*)	3,4,8
Ans	3,4,8
KOrd(*entity*)	4,8
Att(*argument*)	5,6,7,8
FEnc(`sentence`, `target`)	5,6,7,8
FEnc(`sentence`, *argument*)	5,6,7,8
Args(N)	5,6,7,8
KEnc(`target`)	6,7,8
KEnc(*argument*)	6,7,8
Ta(*argument*)	6,7,8
Taa(*argument*, *argument*)	6,7,8
KOrd(`target`)	7,8
KOrd(*argument*)	7,8
FEnc(*argument*, *entity*)	8
Paths(N)	8

setting the feature weights based on annotated training data. Many machine learning algorithms that have been applied to document ranking [4, 12] are well suited to this task. In this work we use an efficient, online linear rank learner, the Committee Perceptron [7].

The Committee Perceptron algorithm is a generalization of previous Perceptron variants [5], and is adapted for learning ranking functions based on preferences between pairs of judged passages. This algorithm significantly outperforms other perceptron variants and performs comparably or better than other linear rank learning algorithms, such as RankSVM, yet requires only a fraction of the training time [7, 12]. Rank learners that minimize the number of mis-ranked passage-pairs such as RankSVM and the Committee Perceptron maximize a lower bound on many common retrieval performance measures such as Mean Average Precision [7]. The algorithm is described in Table 3. Here, we adopt the *indexed sampling* technique proposed by Scully [19] to avoid the quadratic dependence on the number of sentences in the collection.

The passage feature vectors used as input to this algorithm are vectors are represented as: $\mathbf{p}_{iq} = \langle \mathbf{f}_0(p_i, q), \ldots, \mathbf{f}_M(p_i, q) \rangle$ where the $\mathbf{f}_j$; $j = 1 \ldots M$ are the constraint-count features as described in the previous section and f_0 is the

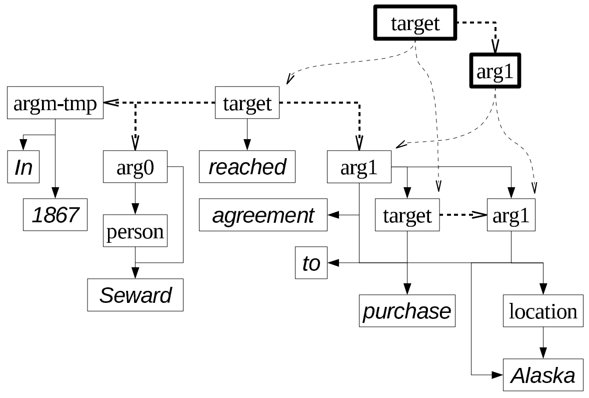

Figure 6: Constraint annotation graph (in bold) aligning to the annotation graph for the answer-bearing passage, *In 1867, ... Seward reached agreement ... to purchase Alaska***. The constraint is "Att(`arg1`)", as shown in Figure 5(4). Note that there are two distinct alignments of the constraint to the passage. The enclosing** `sentence` **and the** *ordering* **relations that exist between** *keyword* **pairs are not shown to increase legibility.**

baseline retrieval score. Features are scaled to zero-mean unit-variance per-question prior to training and testing.

The output of the learning algorithm is a collection of learned weight vectors, $\mathbf{w}^k \in \mathbb{R}^{M+1}; k = 1 \ldots N_{\text{com}}$, and quality indicators for each weight vector, c_k. These weight vectors parameterize an ensemble of linear passage scoring functions: $\text{Score}(\mathbf{p}_{iq}, \mathbf{w}) = \langle \mathbf{p}_{iq}, \mathbf{w} \rangle$ where $\langle \bullet, \bullet \rangle$ is the inner product. Scores from the learned ensemble of linear scoring functions, K, are averaged, weighted by the quality indicator, c_k, to produce the overall score for a passage:

$$\text{Score}^*(\mathbf{p}_{iq}, \mathbf{w}^1, \ldots, \mathbf{w}^{N_{\text{com}}}) = \left\langle \mathbf{p}_{iq}, \frac{\sum_k \mathbf{w}^k \times c_k}{\sum_k c_k} \right\rangle$$

For all experiments, we fix the committee size $N_{\text{com}} = 30$ and the number of passage pairs to sample $T = 10000$, which are known to be effective parameter settings for other tasks.

5. EXPERIMENTAL METHODOLOGY

To evaluate the impact of linguistic and semantic constraint-checking on passage retrieval quality, we introduce the trained ranking model to re-rank the top 1000 sentences retrieved by the Indri baseline, and compare the ranking quality in terms of Mean Average Precision.

5.1 Test Collection

We evaluate on the AQUAINT corpus [11] used in the TREC 2002 QA track, prepared with sentence segmentation (MXTerminator [18]), named entity recognition (BBN Identifinder [1]) and semantic role labeling (ASSERT [17]). The test collection consists of 109[6] factoid questions from the TREC 2002 QA track for which reusable, passage-level relevance judgments are available [13]. The passage-level judgments were aligned to the sentence segmentation on the AQUAINT corpus, excluding relevant passages spanning more than one sentence. Sentences deemed non-answer-bearing due to unresolved anaphora were also excluded.

Information need representations were built from the questions using BBN Identifinder and ASSERT. ASSERT output was hand-corrected to mitigate its poor accuracy on questions, which are relatively rare among its training data. Despite this, only 48 of the 109 information needs (44%) contain ASSERT `targets` and *arguments*[7]. This is because ASSERT explicitly does not cover verbs, including *be*, *have* and *do*, common among the TREC questions. In the analysis, we will examine these 48 *Deep Structure Questions* separately from the remaining 61 *Shallow Structure Questions*.

Table 3: Committee Perceptron Algorithm for Passage Ranking.

```
Input: Number of passage pairs to sample $T$, Committee size $N_{\text{com}}$, List of training relevant/non-relevant passage pairs $S = R \times N = \{(\mathbf{p}_{nq}, \mathbf{p}_{rq})\}$ Output: Set of feature weight vectors and their success counters $K = \{(\mathbf{w}^k, c_k) | k = 1 \ldots N_{\text{com}}\}$
1. Initialize $i = 0$, success counter $c_i = 0$, initial parameters $\mathbf{w}^0$, committee $K = \emptyset$.
2. For $t = 0, \ldots, T$:
      From $S$, sample query $q$ and relevant/non-relevant passages $(\mathbf{p}_{nq}, \mathbf{p}_{rq})$
         If $Score(\mathbf{p}_{nq}, \mathbf{w}^i) \geq Score(\mathbf{p}_{rq}, \mathbf{w}^i)$ then
            $(\mathbf{w}_{\min}, c_{\min}) \in K$ s.t. $c_{\min} = \min_k c_k \in K$
            If $c_i > c_{\min}$ then: add $(\mathbf{w}^i, c_i)$ to $K$
            If $|K| > N_{sub}$: remove $(\mathbf{w}_{\min}, c_{\min})$ from $K$
            update: $\mathbf{w}^{i+1} = \mathbf{w}^i + (\mathbf{p}_{rq} - \mathbf{p}_{nq})$ and $i = i+1$
         Else update: $c_i = c_i + 1$
3. Output: $K$
```

5.2 Feature Groups

As described in Section 4.2, the trained ranking model uses features based on the atomic linguistic and semantic constraints in the information need. Table 2 gives the complete list of features. To study the effectiveness of different types of features, we refer to feature groups numbered 1 through 8, which are described below, in the analysis that follows the experiments.

1. **Keyword Only**: bag-of-words features only.
2. **Surface Patterns**: bag-of-words and relative keyword ordering.
3. **NE + Keywords**: bag-of-words and named entities.
4. **NE + Surface Patterns**: all of the above.

[6]There is some disagreement in the research community as to how much data is required for proper evaluation. One school of thought emphasizes using hundreds of questions and measuring accuracy in terms of matching the TREC-provided answer pattern, as in [20]. Other researchers, such as Moschitti, et. al. [15], have used question sets nearly as small as this one with human relevance judgments. When measuring the quality of a ranked list, we believe that it is more important to have sufficient depth of relevance judgments than it is to have a large number of topics.

[7]Similarly, Shen and Lapata [20] observed that about 35% of TREC factoid questions were answerable using a shallow semantic parsing method based on FrameNet [9].

5. **SRL**: verbs, arguments and semantic roles, ignoring keywords.
6. **SRL + Keywords**: keywords participating in SRL relationships.
7. **SRL + Surface Patterns**: bag-of-words, keyword ordering and SRL.
8. **SRL + NE + Surface Patterns**: all of the above, with arbitrary long-distance semantic relationships between keyword pairs represented by paths through the annotation graph.

6. RESULTS AND DISCUSSION

Experimental results analyzed in this section measure the improvement in passage retrieval quality attributable to re-ranking using linguistic and semantic features, in terms of Mean Average Precision (MAP). Feature groups 1 through 8 are studied individually. For all tests we report the p-value according to the two-sided Fisher's randomization test [21]. First, we describe the results of experiments on the set of all questions, and then we take a closer look at the *Deep Structure Questions*.

6.1 Experiments on the Full Question Set

For the full set of 109 questions, we perform 5-fold cross validation, with approximately 88/22 training/testing queries in each fold. Re-ranking using the trained model yields at least a 5% improvement improvement in MAP for all feature groups. Complete results are given in Table 4.

Table 4: Performance results on the full question set.

	MAP	% over baseline	p-value
Baseline	0.1901		
Feature Group 1	0.2076	9.21	0.0057
2	0.2134	12.26	0.0142
3	0.2142	12.68	0.0036
4	0.2170	14.15	0.0582
5	0.2000	5.21	0.0560
6	0.2157	13.47	0.1171
7	0.2156	13.41	0.1061
8	**0.2329**	22.51	0.0332

In Table 4, most feature groups show noticeable improvements over the baseline, and those of feature groups 1, 2, 3 and 8 are statistically significant. Groups 5, 6 and 7, which use semantic role features, show large, but not significant, gains. Among these data, few questions are helped by the semantic role features, but for those that are, these features can be powerfully predictive. In feature group 6, for example, re-ranking helps 30% of the questions, improving MAP by more than 430% on average, and hurts 26%, degrading quality by 35% on average.

Although these semantic role features do not show significant improvements when training and testing on the whole question set, we do see dramatic improvement on a subset of the questions. We hypothesize that by limiting the training and testing to the *Deep Structure Questions*, we can expect to find larger, more significant improvements in MAP. The *Shallow Structure Questions*, in contrast, have zero feature values for all semantic role labeling features, because ASSERT was not able to provide `target` and *argument* information for these questions.

Table 5: Top 15 (in absolute value) mean feature weights across folds, trained on the full feature set and full question set.

Feature Name	Mean Weight
Ta(`arg1` **)**	203.99
Paths(5)	161.90
Paths(2)	138.04
FEnc(`sentence, date` **)**	128.02
Ans	113.67
KPrec(`sentence` **)**	94.82
Paths(4)	82.90
Baseline Retrieval Score	74.20
KEnc(`date` **)**	57.32
KEnc(`org` **)**	55.52
Paths(1)	-69.22
Taa(`arg1, arg2` **)**	-70.28
KEnc(`person` **)**	-78.54
KPrec(`person` **)**	-96.54
Paths(3)	-180.33

Table 5 shows the 15 features having the greatest magnitude weights learned by the model when training on the full question set, averaged across all five cross validation folds. Though some of the semantic role features claimed the largest magnitude weights, 6 of the 15 most influential features drawn from the surface patterns and named entity feature groups. This is a portrait of the model's difficulty deciding between semantic role features, which are key for some questions yet irrelevant for many, and surface patterns and named entity features, which provide modest help for all questions.

6.2 Deep vs. Shallow Question Structure

To accurately measure the power of the semantic role features, we have to take a more careful look at the *Deep Structure Questions*, as defined in Section 5.1. Table 6 reports the results for the model trained and tested on the *Deep Structure* (top half) and *Shallow Structure* (bottom half) sets separately. As before, these tests show 5-fold cross-validation results averaged across test sets.

From the top half of Table 6, we can clearly see that, for the *Deep Structure Questions*, the semantic role features have a significant positive impact on passage retrieval performance, realizing over 35% improvements in Mean Average Precision when those features are used in combination with keyword information. Feature group 5, which uses semantic role features alone, shows less of an improvement because the semantic role features are most useful when describing linguistic and semantic relationships among keywords that are not implied by surface patterns alone.

For the *Shallow Structure Questions* the picture is vastly different. In the bottom half of Table 6, we observe that the only features that result in moderate improvement are the surface patterns. As expected, feature group 5 shows zero improvement, because for these questions, the feature values other than the Baseline Retrieval Score are always zeroes. What is interesting about these questions is that named entities are of limited use in ranking the passages. It seems that the Baseline Retrieval Score and surface patterns are already capturing most of the information provided by these

Table 6: Performance results on the sets of *Deep* and *Shallow Structure Questions*.

Deep Structure Questions	MAP	% over baseline	p-value
Baseline	0.1978		
Feature Group 1	0.2319	17.24	0.0790
2	0.2176	10.01	0.2167
3	0.2067	4.50	0.3605
4	0.2366	19.62	0.0152
5	0.2159	9.15	0.1269
6	0.2694	36.20	0.0723
7	0.2717	37.36	0.0573
8	**0.2788**	40.95	0.0194

Shallow Structure Questions	MAP	% over baseline	p-value
Baseline	0.1845		
Feature Group 1	0.1835	-0.60	0.5039
2	**0.2014**	9.10	0.0951
3	0.1902	3.03	0.2761
4	0.1864	0.98	0.4619
5	0.1846	0.00	1.0000
6	0.1831	-0.81	0.5097
7	0.1858	0.65	0.4691
8	0.1869	1.25	0.4531

features. Much of the small variations in performance across feature groups 4-7 are attributable to random sampling of the passage-pairs during the training process.

Looking at the features that are assigned the highest weight by the learning algorithm, we see that there is a distinct shift towards strongly favoring semantic role features for the *Deep Structure Questions*. Table 7 shows the top 15 (in absolute value) feature weights learned with the full feature set (8) on the *Deep Structure Questions*, averaged across all five cross validation folds. The top two of these top 15 features encode long-distance linguistic and semantic relationships between keywords that are not implied by the surface representation. Ten of the top 15 make use of some semantic role information. It is interesting to note that the most basic surface pattern features, "Field-keyword-enclosure(`sentence`)" and "Keyword-ordering(`sentence`)", are not a part of the top 15 most useful features, all of which make use of some named entity and/or semantic role information. This fact suggests that semantic role features are indeed powerful for ranking passages with respect to *Deep Structure Questions*.

7. COMPARISON TO STATE-OF-THE-ART APPROACHES

Cui, et. al. [6], suggest a passage ranking method that rewards passages having a dependency tree structure similar to the structure in the question. This similarity score is based on an application of IBM Model 1, which gives the likelihood that the dependency path between two keywords in the answer candidate sentence represents the same syntactic relationship as that which holds between the keyword pair in the question. Cui's method linearly combines this dependency score with a lexical match score from a baseline retrieval method.

Table 7: Top 15 (in absolute value) mean feature weights across folds, trained on the full feature set and the *Deep Structure Questions*.

Feature Name	Weight
Paths(5)	264.45
Paths(4)	211.60
Ans	183.01
Ta(`arg1` **)**	174.95
FEnc(`sentence, date` **)**	99.87
Baseline Retrieval Score	98.42
KEnc(`date` **)**	91.33
KPrec(`arg2` **)**	81.71
Ta(`arg0` **)**	69.97
KEnc(`org` **)**	65.65
KPrec(`arg0` **)**	55.12
Att(`argm-mnr` **)**	-54.89
FEnc(`sentence, argm-mnr` **)**	-54.89
Paths(3)	-66.46
Taa(`arg1, arg2` **)**	-96.68

We conducted a set of experiments on our dataset using the Cui model[8], selecting a mixing weight between the dependency path match score and the lexical score to maximize Mean Average Precision. These experiments showed that our Indri baseline consistently performed as well or better than the Cui model.

The primary finding is that the Cui dependency path match score correlates strongly with the Indri score, with an average per-question Pearson correlation coefficient of 0.7083. See the top row of Figure 7 for a depiction of the correlation between the path match score and the Indri score. Terracing is visible where one score is constant while the other is capturing variation.

The dependency path match score does not appear to offer new information beyond what Indri provides. We hypothesize that it is capturing primarily local syntactic dependencies between phrase heads and their modifiers and between the component words of named entities. Long-distance dependencies are explicitly avoided by Cui, et. al., due to poor parser accuracy [6]. Our baseline Indri queries model local keyword dependencies by enforcing co-occurrence of keywords and named entities of the expected answer type within a small text window.

Moschitti, et. al. [15], proposes a special-purpose tree kernel for the PropBank-style predicate-argument structures used in the type system in this paper. It is shown that the Shallow Semantic Tree Kernel (SSTK) can be used to classify whether an answer candidate is correct given its semantic context and that of the question.

The SSTK maps a predicate-argument structure into a feature space consisting of all possible combinations of `target` verbs and *argument* slots. Each slot can contain at most one keyword, corresponding to the syntactic head of the labeled phrase. SSTK models the question and answer sentences in separate feature spaces, but under our approach, a feature value is jointly derived from a question and an associated answer sentence. Additionally, in contrast to the approach described in this paper, SSTK does not support key-

[8]Experiments were performed using the original code, which is available from: `http://www.cuihang.com/software.html`

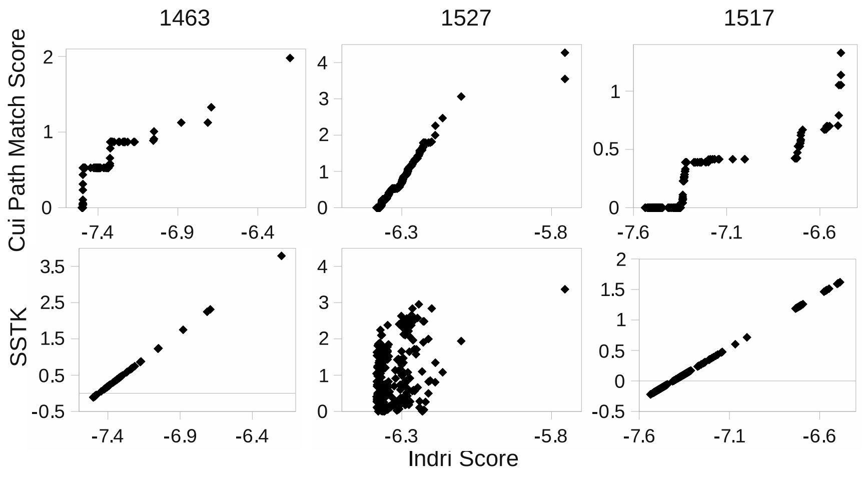

Figure 7: Correlation between Indri score and the Cui Path Match Score (top) and the Moschitti Shallow Semantic Tree Kernel (bottom) for three randomly-selected questions.

word ordering or named entities, long-distance relationships between keywords, or constraints that mix feature types.

SSTK can not be applied directly to our annotation graphs because they are not trees. To compare it to our work, we adapted the kernel function as a feature extractor for use with the Committee Perceptron algorithm. The feature extractor generates all possible combinations of slots from the question and compares them against the answer sentence. Binary feature values are set to one if and only if the answer sentence contains a predicate-argument structure containing the slots filled by the appropriate question keywords.

We repeated the cross-validation experiments using the feature set prescribed by the SSTK, with the addition of the baseline Indri score. We found that the resulting ranking was not statistically significantly better or worse than our Indri baseline. Indeed, with most of the weight accumulating to the Indri score feature, Indri was carrying most of the ranking. The examples in the lower half of Figure 7 show evidence of this, with the Committee Perceptron predictions for some questions correlating near perfectly with Indri and others showing complete confusion. The average per-question Pearson correlation coefficient between the Indri and SSTK scores is 0.6390.

We were not able to reproduce the answer classification results in [15] for several reasons. In the paper, answers are classified for correctness and are ranked by pushing incorrect answers down in an ad hoc fashion. Here, in contrast, we have set up the learning problem as one of pairwise-preference classification, which gives us a complete ranking. The evaluation in the paper focused on description (also known as definition) questions, and our system is optimized for factoid questions.

The SSTK method assumes that a predicate-argument structure has at most one keyword in an argument slot, the syntactic head of the labeled phrase. We speculate that, for the task presented in this paper, the assumption that only the phrasal headwords are important is a bad one. Consider question 1427, *What was the first spaceship on the moon?* Here, the modifier *first* encodes an important constraint that, if ignored, could result in incorrect answers being retrieved.

With both the Cui, et. al., and Moschitti, et. al., methods statistically indistinguishable from our bag-of-words with named entities Indri baseline, we can be confident that our approach demonstrates the power of long-distance linguistic and semantic relationships for passage ranking.

8. RELATED WORK

Rank-learning techniques have been successfully applied to the task of document ranking using features such as term count statistics and baseline retrieval model scores [7, 12]. Recent work applies rank-learning to the task of passage ranking for QA, augmenting these traditional feature types with linguistically-motivated features [23]. Their feature set was based largely on term overlap between the question and answer passage within different syntactic categories, such as verb and subject. A second set of features measured these same overlaps after expansion through WordNet synsets [8].

The Shen and Lapata [20] approach to applying semantics to the task of QA is notable in that it is based on FrameNet [9], rather than PropBank [14], which has a much richer roleset and an inheritance hierarchy for frames. They use a graphical approach to determine the distribution over semantic roles that keywords can take in a question or retrieved sentence. A sentence is assumed to have at most one predicate, chosen heuristically. The similarity function used for ranking answer-bearing sentences requires that the predicates match exactly or participate in the same inheritance chain. The degree of match is determined by the divergence between the role distributions of the arguments of the predicates in the question and answer-bearing sentence.

Both the Verberne, et. al., and Shen, et. al., fail to model the case of multiple predicates in a sentence or question. Furthermore, the Shen, et. al., model does not allow for variation of the predicate verb except along the FrameNet inheritance hierarchy. The method proposed in this paper differs from previous work in that linguistic and semantic constraints between keyword pairs are modeled directly. Keywords labeled with semantic roles can be associated with specific target verbs in the event that there are more than one in the sentence. In addition, our approach models the case of multiple *arguments* attached to an unknown target.

Single-pass passage retrieval methods have also been proposed for QA, which involve pre-annotating the collection and indexing linguistic and semantic features to enable query-time constraint-checking. Bilotti, et. al., use structured queries to retrieve text satisfying PropBank-style semantic constraints [3]. Their method is shown to be effective in certain cases, yet poor at combining evidence from bag-of-words and structured features and not robust to ranking partial matches. Pizzato and Mollá apply the vector-space model to a feature space consisting of surface words and their semantic role labels, as well as pairwise semantic relationships between keywords [16]. This model performs better at partial matching, but does not capture long-distance linguistic and semantic relationships between keywords.

9. CONTRIBUTIONS

For Question Answering (QA) systems to ever hope to compete with the dominant web search information access paradigm, they must improve both speed and accuracy. Poor quality embedded passage retrieval is one of the primary causes of wrong answers and high latency as perceived by users. This work aims to address the issue of passage retrieval quality, proposing a novel passage ranking framework for QA that unifies traditional keyword-only ranking features with deeper linguistic and semantic features in a rank learning framework. Experimental results show that this generalized linguistic and semantic text similarity approach significantly outperforms a high-quality bag-of-words and named entity passage retrieval baseline on common classes of factoid questions, achieving better than a 20% increase in Mean Average Precision (MAP).

For questions analyzable by ASSERT, the improvement in MAP offered by the proposed method can reach 40%. Currently, fewer than half of the questions in the test collection can be analyzed by ASSERT. Despite the small sample size, the 40% improvement in MAP is statistically significant. For the questions that do not have ASSERT analyses, average precision is not significantly penalized, so there is little risk in deploying the proposed passage retrieval method. As coverage is improved for semantic role labeling tools such as ASSERT, a greater percentage of user's questions will be analyzable, which will result in even greater improvements in passage retrieval quality for a QA system employing the proposed method. The future potential for the proposed approach notwithstanding, even the 20% improvement in MAP available using today's NLP tools could make a substantial impact on the QA research community by enabling faster and more accurate Answer Generation that could result in lower end-to-end system latency and better accuracy that users would notice.

10. ACKNOWLEDGEMENTS

This material is based in part upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract Number W0853736. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

11. REFERENCES

[1] D. Bikel, R. Schwartz, and R. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1–3):211–231, 1999.

[2] M. W. Bilotti. *Linguistic and Semantic Passage Retrieval Strategies for Question Answering.* PhD thesis, Carnegie Mellon University, 2009.

[3] M. W. Bilotti, P. Ogilvie, J. Callan, and E. Nyberg. Structured retrieval for question answering. In *Proc. SIGIR*, 2007.

[4] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proc. ICML*, 2005.

[5] M. Collins. Ranking algorithms for named-entity extraction: boosting and the voted perceptron. In *Proc. ACL*, 2001.

[6] H. Cui, R. Sun, K. Li, M.-Y. Kan, and T.-S. Chua. Question answering passage retrieval using dependency relations. In *Proc. SIGIR*, 2005.

[7] J. L. Elsas, V. R. Carvalho, and J. G. Carbonell. Fast learning of document ranking functions with the committee perceptron. In *Proc. WSDM*, 2008.

[8] C. Fellbaum. *WordNet, an Electronic Lexical Database.* MIT Press, Cambridge, MA, 1998.

[9] C. Fillmore, C. Johnson, and M. Petruck. Background to framenet. *International Journal of Lexicography*, 16:235–250, 2003.

[10] X. Geng, T. Liu, T. Qin, and H. Li. Feature selection for ranking. In *Proc. SIGIR*, 2007.

[11] D. Graff. *The AQUAINT Corpus of English News Text.* Linguistic Data Consortium, 2002. Cat. No. LDC2002T31.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. KDD*, 2002.

[13] M. Kaisser and J. B. Lowe. Creating a research collection of question answer sentence pairs with amazon's mechanical turk. In *Proc. LREC*, 2008.

[14] P. Kingsbury, M. Palmer, and M. Marcus. Adding semantic annotation to the penn treebank. In *Proc. HLT*, 2002.

[15] A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *Proc. ACL*, 2007.

[16] L. A. Pizzato and D. Mollá. Indexing on semantic roles for question answering. In *Proc. IR4QA Workshop at COLING*, 2008.

[17] S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. Shallow semantic parsing using support vector machines. In *Proc. HLT*, 2004.

[18] J. Reynar and A. Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proc. ANLP*, 1997.

[19] D. Sculley. Large scale learning to rank. In *NIPS 2009 Workshop on Advances in Ranking*, 2009.

[20] D. Shen and M. Lapata. Using semantic roles to improve question answering. In *Proc. EMNLP*, 2007.

[21] M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proc. CIKM*, 2007.

[22] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proc. SIGIR*, 2003.

[23] S. Verberne, H. van Halteren, D. Theijssen, S. Raaijmakers, and L. Boves. Learning to rank qa data. In *Proc. LR4IR Workshop at SIGIR*, 2009.