Learning to Rank Answers to Non-Factoid Questions from Web Collections

Mihai Surdeanu*
Stanford University

Massimiliano Ciaramita**
Google Inc.

Hugo Zaragoza†
Yahoo! Research

This work investigates the use of linguistically motivated features to improve search, in particular for ranking answers to non-factoid questions. We show that it is possible to exploit existing large collections of question–answer pairs (from online social Question Answering sites) to extract such features and train ranking models which combine them effectively. We investigate a wide range of feature types, some exploiting natural language processing such as coarse word sense disambiguation, named-entity identification, syntactic parsing, and semantic role labeling. Our experiments demonstrate that linguistic features, in combination, yield considerable improvements in accuracy. Depending on the system settings we measure relative improvements of 14% to 21% in Mean Reciprocal Rank and Precision@1, providing one of the most compelling evidence to date that complex linguistic features such as word senses and semantic roles can have a significant impact on large-scale information retrieval tasks.

1. Introduction

The problem of Question Answering (QA) has received considerable attention in the past few years. Nevertheless, most of the work has focused on the task of factoid QA, where questions match short answers, usually in the form of named or numerical entities. Thanks to international evaluations organized by conferences such as the Text REtrieval Conference (TREC) and the Cross Language Evaluation Forum (CLEF) Workshop, annotated corpora of questions and answers have become available for several languages, which has facilitated the development of robust machine learning models for the task.[1]

* Stanford University, 353 Serra Mall, Stanford, CA 94305–9010. E-mail: mihais@stanford.edu.
** Google Inc., Brandschenkestrasse 110, CH–8002 Zürich, Switzerland. E-mail: massi@google.com.
† Yahoo! Research, Avinguda Diagonal 177, 8th Floor, 08018 Barcelona, Spain. E-mail: hugoz@yahoo-inc.com.

1 TREC: http://trec.nist.gov; CLEF: http://www.clef-campaign.org.

The primary part of this work was carried out while all authors were working at Yahoo! Research.

Submission received: 1 April 2010; revised submission received: 11 September 2010; accepted for publication: 23 November 2010.

Table 1
Sample content from Yahoo! Answers.

High Quality	*Q*: How do you quiet a squeaky door? *A*: Spray WD-40 directly onto the hinges of the door. Open and close the door several times. Remove hinges if the door still squeaks. Remove any rust, dirt or loose paint. Apply WD-40 to removed hinges. Put the hinges back, open and close door several times again.
High Quality	*Q*: How does a helicopter fly? *A*: A helicopter gets its power from rotors or blades. So as the rotors turn, air flows more quickly over the tops of the blades than it does below. This creates enough lift for flight.
Low Quality	*Q*: How to extract html tags from an html documents with c++? *A*: very carefully

The situation is different once one moves beyond the task of factoid QA. Comparatively little research has focused on QA models for non-factoid questions such as causation, manner, or reason questions. Because virtually no training data is available for this problem, most automated systems train either on small hand-annotated corpora built in-house (Higashinaka and Isozaki 2008) or on question–answer pairs harvested from Frequently Asked Questions (FAQ) lists or similar resources (Soricut and Brill 2006; Riezler et al. 2007; Agichtein et al. 2008). None of these situations is ideal: The cost of building the training corpus in the former setup is high; in the latter scenario the data tend to be domain-specific, hence unsuitable for the learning of open-domain models, and for drawing general conclusions about the underlying scientific problems.

On the other hand, recent years have seen an explosion of user-generated content (or social media). Of particular interest in our context are community-driven question-answering sites, such as Yahoo! Answers, where users answer questions posed by other users and best answers are selected manually either by the asker or by all the participants in the thread.[2] The data generated by these sites have significant advantages over other Web resources: (a) they have a high growth rate and they are already abundant; (b) they cover a large number of topics, hence they offer a better approximation of open-domain content; and (c) they are available for many languages. Community QA sites, similar to FAQs, provide a large number of question–answer pairs. Nevertheless, these data have a significant drawback: they have high variance of quality (i.e., questions and answers range from very informative to completely irrelevant or even abusive). Table 1 shows some examples of both high and low quality content from the Yahoo! Answers site.

In this article we investigate two important aspects of non-factoid QA:

1. *Is it possible to learn an answer-ranking model for non-factoid questions, in a completely automated manner, using data available in on-line social QA sites?* This is an interesting question because a positive answer indicates that a plethora of training data are readily available to researchers and system

2 `http://answers.yahoo.com`.

developers working on natural language processing, information retrieval, and machine learning.

2. *Which features and models are more useful in this context, that is, ample but noisy data?* For example: Are similarity models as effective as models that learn question-to-answer transformations? Does syntactic and semantic information help?

Social QA sites are the ideal vehicle to investigate such questions. Questions posted on these sites typically have a correct answer that is selected manually by users. Assuming that all the other candidate answers are incorrect (we discuss this assumption in Section 5), it is trivial to automatically organize these data into a format ready for discriminative learning, namely, the pair question–correct answer generates one positive example and all other answers for the same question are used to generate negative examples. This allows one to use the collection in a completely automated manner to learn answer ranking models.

The contributions of our investigation are the following:

1. We introduce and evaluate many linguistic features for answer re-ranking. Although several of these features have been introduced in previous work, some are novel in the QA context, for example, syntactic dependencies and semantic role dependencies with words generalized to semantic tags. Most importantly, to the best of our knowledge this is the first work that combines all these features into a single framework. This allows us to investigate their comparative performance in a formal setting.
2. We propose a simple yet powerful representation for complex linguistic features, that is, we model syntactic and semantic information as bags of syntactic dependencies or semantic role dependencies and build similarity and translation models over these representations. To address sparsity, we incorporate a back-off approach by adding additional models where lexical elements in these structures are generalized to semantic tags. These models are not only simple to build, but, as our experiments indicate, they perform at least as well as complex, dedicated models such as tree kernels.
3. We are the first to evaluate the impact of such linguistic features in a large-scale setting that uses real-world noisy data. The impact on QA of some of the features we propose has been evaluated before, but these experiments were either on editorialized data enhanced with gold semantic structures (e.g., the Wall Street Journal corpus with semantic roles from PropBank [Bilotti et al. 2007]), or on very few questions (e.g., 413 questions from TREC 12 [Cui et al. 2005]). On the other hand, we evaluate on over 25,000 questions, and each question has up to 100 candidate answers from Yahoo! Answers. All our data are processed with off-the-shelf natural language (NL) processors.

The article is organized as follows. We describe our approach, including all the features explored for answer modeling, in Section 2. We introduce the corpus used in our empirical analysis in Section 3. We detail our experiments and analyze the results

in Section 4. Section 5 discusses current shortcomings of our system and proposes solutions. We overview related work in Section 6 and conclude the article in Section 7.

2. Approach

Figure 1 illustrates our QA architecture. The processing flow is the following. First, the **answer retrieval** component extracts a set of candidate answers **A** for a question Q from a large collection of answers, **C**, provided by a community-generated question-answering site. The retrieval component uses a state-of-the-art information retrieval (IR) model to extract **A** given Q. The second component, **answer ranking**, assigns to each answer $A_i \in \mathbf{A}$ a score that represents the likelihood that A_i is a correct answer for Q, and ranks all answers in descending order of these scores. In our experiments, the collection **C** contains all answers previously selected by users of a social QA site as best answers for non-factoid questions of a certain type (e.g., "How to" questions). The entire collection of questions, **Q**, is split into a training set and two held-out sets: a development one used for parameter tuning, and a testing one used for the formal evaluation.

Our architecture follows closely the architectures proposed in the TREC QA track (see, e.g., Voorhees 2001). For efficiency reasons, most participating systems split the answer extraction phase into a retrieval phase that selected likely answer snippets using shallow techniques, followed by a (usually expensive) answer ranking phase that processes only the candidates proposed by the retrieval component. Due to this separation, such architectures can scale to collections of any size. We discuss in Section 6 how related work has improved this architecture further—for example, by adding query expansion terms from the translation models back to answer retrieval (Riezler et al. 2007).

The focus of this work, however, is on the re-ranking model implemented in the answer ranking component. We call this model FMIX—from **f**eature **mix**—because the proposed scoring function is a linear combination of four different classes of features

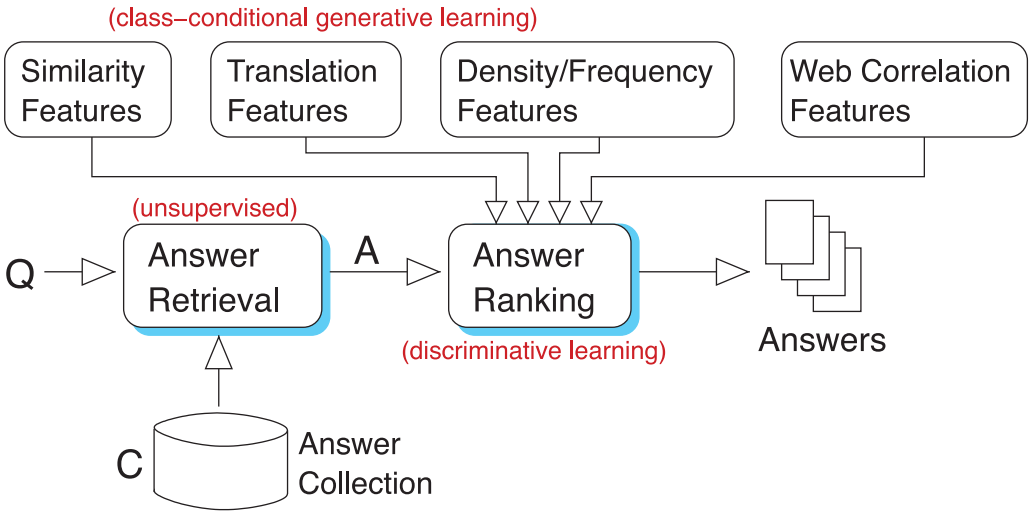

Figure 1
Architecture of our QA framework.

(detailed in Section 2.2). To accommodate and combine all these feature classes, our QA approach combines three types of machine learning methodologies (as highlighted in Figure 1): the answer retrieval component uses unsupervised IR models, the answer ranking is implemented using discriminative learning, and finally, some of the ranking features are produced by question-to-answer translation models, which use class-conditional generative learning. To our knowledge, this combined approach is novel in the context of QA. In the remainder of the article, we will use the FMIX function to answer the research objectives outlined in the Introduction. To answer the first research objective we will compare the quality of the rankings provided by this component against the rankings generated by the IR model used for answer retrieval. To answer the second research objective we will analyze the contribution of the proposed feature set to this function.

We make some simplifying assumptions in this study. First, we will consider only manner questions, and in particular only "How to" questions. This makes the corpus more homogeneous and more focused on truly informational questions (as opposed to social questions such as "Why don't girls like me?", or opinion questions such as "Who will win the next election?", both of which are very frequent in Yahoo! Answers). Second, we concentrate on the task of answer-re-ranking, and ignore all other modules needed in a complete on-line social QA system. For example, we ignore the problem of matching questions to questions, very useful when retrieving answers in a FAQ or a QA collection (Jeon, Croft, and Lee 2005), and we ignore all "social" features such as the authority of users (Jeon et al. 2006; Agichtein et al. 2008). Instead, we concentrate on matching answers and on the different textual features. Hence, the document collection used in our experiments contains only answers, without the corresponding questions answered. Furthermore, we concentrate on the re-ranking phase and we do not explore techniques to improve the recall of the initial retrieval phase (by methods of query expansion, for example). Such aspects are complementary to our work, and can be investigated separately.

2.1 Representations of Content

One of our main interests in using very large data sets was to show that complex linguistic features can improve ranking models if they are correctly combined with simpler features, in particular using discriminative learning methods on a particular task. For this reason we explore several forms of textual representation going beyond the bag of words. In particular, we generate our features over four different representations of text:

Words (W): This is the traditional IR view where the text is seen as a bag of words.

***n*-grams (N):** The text is represented as a bag of word *n*-grams, where *n* ranges from two up to a given length (we discuss structure parameters in the following).

Dependencies (D): The text is converted to a bag of syntactic dependency chains. We extract syntactic dependencies in the style of the CoNLL-2007 shared task using the syntactic processor described in Section 3.[3] From the tree of syntactic dependencies we extract all the paths up to a given length following modifier-to-head links. The top part

3 http://depparse.uvt.nl/depparse-wiki/SharedTaskWebsite.

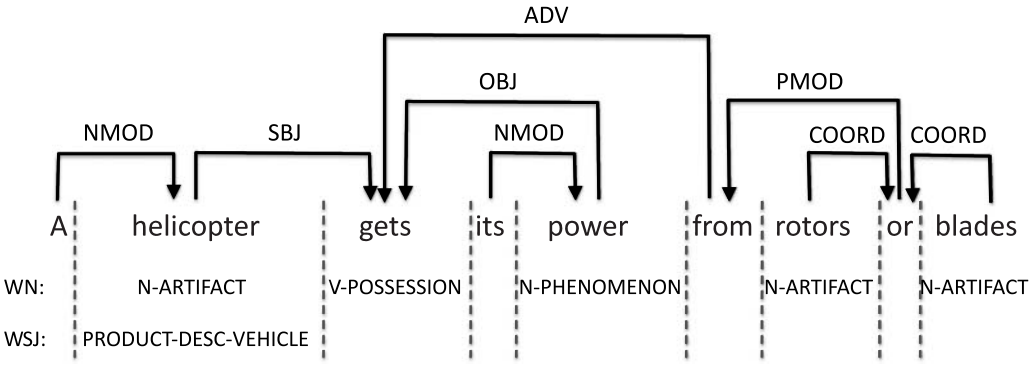

Figure 2
Sample syntactic dependencies and semantic tags.

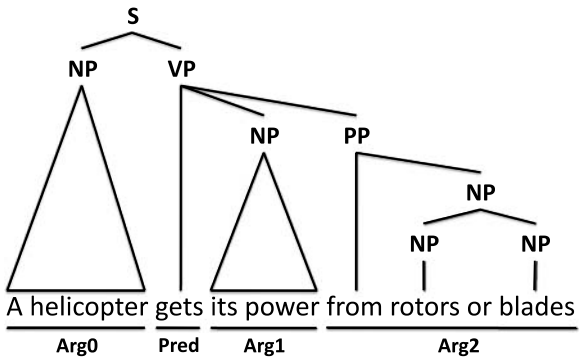

Figure 3
Sample semantic proposition.

of Figure 2 shows a sample corpus sentence with the actual syntactic dependencies extracted by our syntactic processor. The figure indicates that this representation captures important syntactic relations, such as subject–verb (e.g., *helicopter* $\xrightarrow{\text{SBJ}}$ *gets*) or object-verb (e.g., *power* $\xrightarrow{\text{OBJ}}$ *gets*).

Semantic Roles (R): The text is represented as a bag of predicate–argument relations extracted using the semantic parser described in Section 3. The parser follows the PropBank notations (Palmer, Gildea, and Kingsbury 2005), that is, it assigns semantic argument labels to nodes in a constituent-based syntactic tree. Figure 3 shows an example. The figure shows that the semantic proposition corresponding to the predicate *gets* includes *A helicopter* as the `Arg0` argument (`Arg0` stands for agent), *its power* as the `Arg1` argument (or patient), and *from rotors or blades* as `Arg2` (or instrument). Semantic roles have the advantage that they extract meaning beyond syntactic representations (e.g., a syntactic subject may be either an agent or a patient in the actual proposition). We convert the semantic propositions detected by our parser into semantic dependencies using the same approach as Surdeanu et al. (2008), that is, we create a semantic dependency between each predicate and the syntactic head of every one of its arguments. These dependencies are labeled with the label of the corresponding argument. For example, the semantic dependency that includes the `Arg0` argument in Figure 3 is represented as *gets* $\xrightarrow{\texttt{Arg0}}$ *helicopter*. If the syntactic constituent corresponding to a semantic argument is

a prepositional phrase (PP), we convert it to a bigram that includes the preposition and the head word of the attached phrase. For example, the tuple for Arg2 in the example is represented as *gets* $\xrightarrow{\texttt{Arg2}}$ *from-rotors*.

In all representations we remove structures where either one of the elements is a stop word and convert the remaining words to their WordNet lemmas.[4]

The structures we propose are highly configurable. In this research, we investigate this issue along three dimensions:

Degree of lexicalization: We reduce the sparsity of the proposed structures by replacing the lexical elements with semantic tags which might provide better generalization. In this article we use two sets of tags, the first consisting of coarse WordNet senses, or supersenses (WNSS) (Ciaramita and Johnson 2003), and the second of named-entity labels extracted from the Wall Street Journal corpus. We present in detail the tag sets and the processors used to extract them in Section 3. For an overview, we show a sample annotated sentence in the bottom part of Figure 2.

Labels of relations: Both dependency and predicate–argument relations can be labeled or unlabeled (e.g., *gets* $\xrightarrow{\texttt{Arg0}}$ *helicopter* versus *gets* → *helicopter*). We make this distinction in our experiments for two reasons: (a) removing relation labels reduces the model sparsity because fewer elements are created, and (b) performing relation recognition without classification is simpler than performing the two tasks, so the corresponding NL processors might be more robust in the unlabeled-relation setup.

Structure size: This parameter controls the size of the generated structures, namely, number of words in *n*-grams or dependency chains, or number of elements in the predicate–argument tuples. Nevertheless, in our experiments we did not see any improvements from structure sizes larger than two. In the experiments reported in this article, all the structures considered are of size two, that is, we use bigrams, dependency chains of two elements, and tuples of one predicate and one semantic argument.

2.2 Features

We explore a rich set of features inspired by several state-of-the-art QA systems (Harabagiu et al. 2000; Magnini et al. 2002; Cui et al. 2005; Soricut and Brill 2006; Bilotti et al. 2007; Ko, Mitamura, and Nyberg 2007). To the best of our knowledge this is the first work that: (a) adapts all these features for non-factoid answer ranking, (b) combines them in a single scoring model, and (c) performs an empirical evaluation of the different feature families and their combinations.

For clarity, we group the features into four sets: features that model the similarity between questions and answers (FG1), features that encode question-to-answer transformations using a translation model (FG2), features that measure keyword density and frequency (FG3), and features that measure the correlation between question–answer pairs and other collections (FG4). Wherever applicable, we explore different syntactic and semantic representations of the textual content, as introduced previously. We next explain in detail each of these feature groups.

4 http://wordnet.princeton.edu.

FG1: Similarity Features. We measure the similarity between a question Q and an answer A using the length-normalized BM25 formula (Robertson and Walker 1997), which computes the score of the answer A as follows:

$$BM25(A) = \sum_{i=0}^{|Q|} \frac{(k_1+1)tf_i^A(k_3+1)tf_i^Q}{(K+tf_i^A)(k_3+tf_i^Q)} \log(idf_i) \quad (1)$$

where tf_i^A and tf_i^Q are the frequencies of the question term i in A and Q, and idf_i is the inverse document frequency of term i in the answer collection. K is the length-normalization factor:

$$K = k_1((1-b) + b|A|/avg\_len)$$

where avg_len is the average answer length in the collection. For all the constants in the formula (b, k_1, and k_3) we use values reported optimal for other IR collections ($b = 0.75$, $k_1 = 1.2$, and $k_3 = 1,000$).

We chose this similarity formula because, of all the IR models we tried, it provided the best ranking at the output of the answer retrieval component. For completeness we also include in the feature set the value of the $tf \cdot idf$ similarity measure. For both formulas we use the implementations available in the Terrier IR platform with the default parameters.[5]

To understand the contribution of our syntactic and semantic processors we compute the similarity features for different representations of the question and answer content, ranging from bag of words to semantic roles. We detail these representations in Section 2.1.

FG2: Translation Features. Berger et al. (2000) showed that similarity-based models are doomed to perform poorly for QA because they fail to "bridge the lexical chasm" between questions and answers. One way to address this problem is to learn question-to-answer transformations using a translation model (Berger et al. 2000; Echihabi and Marcu 2003; Soricut and Brill 2006; Riezler et al. 2007). In our model, we incorporate this approach by adding the probability that the question Q is a translation of the answer A, $P(Q|A)$, as a feature. This probability is computed using IBM's Model 1 (Brown et al. 1993):

$$P(Q|A) = \prod_{q \in Q} P(q|A) \quad (2)$$

$$P(q|A) = (1-\lambda)P_{ml}(q|A) + \lambda P_{ml}(q|\mathbf{C}) \quad (3)$$

$$P_{ml}(q|A) = \sum_{a \in A} (T(q|a)P_{ml}(a|A)) \quad (4)$$

where the probability that the question term q is generated from answer A, $P(q|A)$, is smoothed using the prior probability that the term q is generated from the entire collection of answers $\mathbf{C}$, $P_{ml}(q|\mathbf{C})$. λ is the smoothing parameter. $P_{ml}(q|\mathbf{C})$ is computed

5 http://ir.dcs.gla.ac.uk/terrier.

using the maximum likelihood estimator. To mitigate sparsity, we set $P_{ml}(q|\mathbf{C})$ to a small value for out-of-vocabulary words.[6] $P_{ml}(q|A)$ is computed as the sum of the probabilities that the question term q is a translation of an answer term a, $T(q|a)$, weighted by the probability that a is generated from A. The translation table for $T(q|a)$ is computed using the EM algorithm implemented in the GIZA++ toolkit.[7]

Translation models have one important limitation when used for retrieval tasks: They do not guarantee that the probability of translating a word to itself, that is, $T(w|w)$, is high (Murdock and Croft 2005). This is a problem for QA, where word overlap between question and answer is a good indicator of relevance (Moldovan et al. 1999). We address this limitation with a simple algorithm: we set $T(w|w) = 0.5$ and re-scale the other $T(w'|w)$ probabilities for all other words w' in the vocabulary to sum to 0.5, to guarantee that $\sum_{w'} T(w'|w) = 1$. This has the desired effect that $T(w|w)$ becomes larger than any other $T(w'|w)$. Our initial experiments proved empirically that this is essential for good performance.

As prior work indicates, tuning the smoothing parameter λ is also crucial for the performance of translation models, especially in the context of QA (Xue, Jeon, and Croft 2008). We tuned the λ parameter independently for each of the translation models introduced as follows: (a) for a smaller subset of the development corpus introduced in Section 3 (1,500 questions) we retrieved candidate answers using our best retrieval model (BM25); (b) we implemented a simple re-ranking model using as the only feature the translation model probability; and (c) we explored a large range of values for λ and selected the one that maximizes the mean reciprocal rank (MRR) of the re-ranking model. This process selected a wide range of values for the λ parameter for the different translation models (e.g., 0.09 for the translation model over labeled syntactic dependencies, and 0.43 for the translation model over labeled semantic role dependencies).

Similarly to the previous feature group, we add translation-based features for the different text representations detailed in Section 2.1. By moving beyond the bag-of-words representation we hope to learn relevant transformations of structures, for example, from the *squeaky* → *door* dependency to *spray* ← *WD-40* in the Table 1 example.

FG3: Density and Frequency Features. These features measure the density and frequency of question terms in the answer text. Variants of these features were used previously for either answer or passage ranking in factoid QA (Moldovan et al. 1999; Harabagiu et al. 2000). Tao and Zhai (2007) evaluate a series of proximity-based measures in the context of information retrieval.

Same word sequence: Computes the number of non-stop question words that are recognized in the same order in the answer.

Answer span: The largest distance (in words) between two non-stop question words in the answer. We compute multiple variants of this feature, where we count: (a) the total number of non-stop words in the span, or (b) the number of non-stop nouns.

Informativeness: Number of non-stop nouns, verbs, and adjectives in the answer text that do not appear in the question.

6 We used 1E-9 for the experiments in this article.
7 `http://www.fjoch.com/GIZA++.html`.

Same sentence match: Number of non-stop question terms matched in a single sentence in the answer. This feature is added both unnormalized and normalized by the question length.

Overall match: Number of non-stop question terms matched in the complete answer.

All these features are computed as raw counts and as normalized counts (dividing the count by the question length, or by the answer length in the case of **Answer span**). The last two features (**Same sentence match** and **Overall match**) are computed for all text representations introduced, including syntactic and semantic dependencies (see Section 2.1).

Note that counting the number of matched syntactic dependencies is essentially a simplified tree kernel for QA (e.g., see Moschitti et al. 2007) matching only trees of depth 2. We also include in this feature group the following tree-kernel features.

Tree kernels: To model larger syntactic structures that are shared between questions and answers we compute the tree kernel values between all question and answer sentences. We implemented a dependency-tree kernel based on the convolution kernels proposed by Collins and Duffy (2001). We add as features the largest value measured between any two individual sentences, as well as the average of all computed kernel values for a given question and answer. We compute tree kernels for both labeled and unlabeled dependencies, and for both lexicalized trees and for trees where words are generalized to their predicted WNSS or named-entity tags (when available).

FG4: Web Correlation Features. Previous work has shown that the redundancy of a large collection (e.g., the Web) can be used for answer validation (Brill et al. 2001; Magnini et al. 2002). In the same spirit, we add features that measure the correlation between question–answer pairs and large external collections:

Web correlation: We measure the correlation between the question–answer pair and the Web using the Corrected Conditional Probability (CCP) formula of Magnini et al. (2002):

$$CCP(Q, A) = hits(Q + A)/(hits(Q)\, hits(A)^{2/3}) \quad (5)$$

where *hits* returns the number of page hits from a search engine. The *hits* procedure constructs a Boolean query from the given set of terms, represented as a conjunction of all the corresponding keywords. For example, for the second question in Table 1, $hits(Q)$ uses the Boolean query: *helicopter* `AND` *fly*.

It is notable that this formula is designed for Web-based QA, that is, the conditional probability is adjusted with $1/hits(A)^{2/3}$ to reduce the number of cases when snippets containing high-frequency words are marked as relevant answers. This formula was shown to perform best for the task of QA (Magnini et al. 2002). Nevertheless, this formula was designed for factoid QA, where both the question and the exact answer have a small number of terms. This is no longer true for non-factoid QA. In this context it is likely that the number of hits returned for Q, A, or $Q + A$ is zero given the large size of the typical question and answer. To address this issue, we modified the *hits* procedure to include a simple iterative query relaxation algorithm:

1. Assign keyword priorities using a set of heuristics inspired by Moldovan et al. (1999). The complete priority detection algorithm is listed in Table 2.

Table 2
Keyword priority heuristics.

Step	Keyword type	Priority
(a)	Non-stop keywords within quotes	8
(b)	Non-stop keywords tagged as proper nouns	7
(c)	Contiguous sequences of 2+ adjectives as nouns	6
(d)	Contiguous sequences of 2+ nouns	5
(e)	Adjectives not assigned in step (c)	4
(f)	Nouns not assigned in steps (c) or (d)	3
(g)	Verbs and adverbs	2
(h)	Non-stop keywords not assigned in the previous steps	1

2. Fetch the number of page hits using the current query.
3. If the number of hits is larger than zero, stop; otherwise discard the set of keywords with the smallest priority in the current query and repeat from step 2.

Query-log correlation: As in Ciaramita, Murdock, and Plachouras (2008), we also compute the correlation between question–answer pairs from a search-engine query-log corpus of more than 7.5 million queries, which shares roughly the same time stamp with the community-generated question–answer corpus. Using the query-log correlation between two snippets of text was shown to improve performance for contextual advertising, that is, linking a user's query to the description of an ad (Ciaramita, Murdock, and Plachouras 2008). In this work, we adapt this idea to the task of QA. However, because it is not clear which correlation metric performs best in this context, we compute both the Pointwise Mutual Information (PMI) and chi square (χ^2) association measures between each question–answer word pair in the query-log corpus. The largest and the average values are included as features, as well as the number of QA word pairs which appear in the top 10, 5, and 1 percentile of the PMI and χ^2 word pair rankings.

We replicate all features that can be computed for different content representations using every independent representation and parameter combination introduced in Section 2.1. For example, we compute similarity scores (FG1) for 16 different representations of question/answer content, produced by different parametrizations of the four different generic representations (W, N, D, R). One important exception to this strategy are the translation-model features (FG2). Because our translation models aim to learn both lexical and structural transformations between questions and answers, it is important to allow structural variations in the question/answer representations. In this article, we implement a simple and robust approximation for this purpose: For translation models we concatenate all instances of structured representations (N, D, R) with the corresponding bag-of-words representation (W). This allows the translation models to learn some combined lexical and structural transformation (e.g., from the dependency *squeaky* → *door* dependency to the token *WD-40*). All in all, replicating our features for all the different content representations yields 137 actual features to be used for learning.

2.3 Ranking Models

Our approach is agnostic with respect to the actual learning model. To emphasize this, we experimented with two learning algorithms. First, we implemented a variant of the ranking Perceptron proposed by Shen and Joshi (2005). In this framework the ranking problem is reduced to a binary classification problem. The general idea is to exploit the pairwise preferences induced from the data by training on pairs of patterns, rather than independently on each pattern. Given a weight vector α, the score for a pattern $\mathbf{x}$ (a candidate answer) is given by the inner product between the pattern and the weight vector:

$$f_\alpha(\mathbf{x}) = \langle \mathbf{x}, \alpha \rangle \tag{6}$$

However, the error function depends on pairwise scores. In training, for each pair $(\mathbf{x}_i, \mathbf{x}_j) \in \mathbf{A}$, the score $f_\alpha(\mathbf{x}_i - \mathbf{x}_j)$ is computed; note that if f is an inner product $f_\alpha(\mathbf{x}_i - \mathbf{x}_j) = f_\alpha(\mathbf{x}_i) - f_\alpha(\mathbf{x}_j)$. In this framework one can define suitable margin functions that take into account different levels of relevance; for example, Shen and Joshi (2005) propose $g(i,j) = (\frac{1}{i} - \frac{1}{j})$, where i and j are the rank positions of x_i and x_j. Because in our case there are only two relevance levels we use a simpler sign function $y_{i,j}$, which is negative if $i > j$ and positive otherwise; $y_{i,j}$ is then scaled by a positive rate τ found empirically on the development data. In the presence of numbers of possible rank levels appropriate margin functions can be defined. During training, if $f_\alpha(\mathbf{x}_i - \mathbf{x}_j) \leq y_{i,j}\tau$, an update is performed as follows:

$$\alpha^{t+1} = \alpha^t + (\mathbf{x}_i - \mathbf{x}_j) y_{i,j} \tau \tag{7}$$

We notice, in passing, that variants of the perceptron including margins have been investigated before; for example, in the context of uneven class distributions (see Li et al. 2002). It is interesting to notice that such variants have been found to be competitive with SVMs in terms of performance, while being more efficient (Li et al. 2002; Surdeanu and Ciaramita 2007). The comparative evaluation from our experiments are consistent with these findings. For regularization purposes, we use as a final model the average of all Perceptron models posited during training (Freund and Schapire 1999).

We also experimented with SVM-rank (Joachims 2006), which is an instance of structural SVM—a family of Support Vector Machine algorithms that model structured outputs (Tsochantaridis et al. 2004)—specifically tailored for ranking problems.[8] SVM-rank optimizes the area under a ROC curve. The ROC curve is determined by the true positive rate vs. the false positive rate for varying values of the prediction threshold, thus providing a metric closely related to Mean Average Precision (MAP).

3. The Corpus

The corpus is extracted from a sample of the U.S. Yahoo! Answers questions and answers. We focus on the subset of advice or "how to" questions due to their frequency, quality, and importance in social communities. Nevertheless, our approach

8 http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html.

is independent of the question type. To construct our corpus, we implemented the following successive filtering steps:

Step 1: From the full corpus we keep only questions that match the regular expression:
`how (to|do|did|does|can|would|could|should)`
and have an answer selected as best either by the asker or by the participants in the thread. The outcome of this step is a set of 364,419 question–answer pairs.

Step 2: From this corpus we remove the questions and answers of dubious quality. We implement this filter with a simple heuristic by keeping only questions and answers that have at least four words each, out of which at least one is a noun and at least one is a verb. The rationale for this step is that genuine answers to "how to" questions should have a minimal amount of structure, approximated by the heuristic. This step filters out questions like *How to be excellent?* and answers such as *I don't know*. The outcome of this step forms our answer collection **C**. **C** contains 142,627 question–answer pairs. This corpus is freely available through the Yahoo! Webscope program.[9]

Arguably, all these filters could be improved. For example, the first step can be replaced by a question classifier (Li and Roth 2006). Similarly, the second step can be implemented with a statistical classifier that ranks the quality of the content using both the textual and non-textual information available in the database (Jeon et al. 2006; Agichtein et al. 2008). We plan to further investigate these issues, which are not the main object of this work.

The data was processed as follows. The text was split at the sentence level, tokenized and POS tagged, in the style of the Wall Street Journal Penn TreeBank (Marcus, Santorini, and Marcinkiewicz 1993). Each word was morphologically simplified using the morphological functions of the WordNet library. Sentences were annotated with WNSS categories, using the tagger of Ciaramita and Altun (2006), which annotates text with a 46-label tagset.[10] These tags, defined by WordNet lexicographers, provide a broad semantic categorization for nouns and verbs and include labels for nouns such as food, animal, body, and feeling, and for verbs labels such as communication, contact, and possession. We chose to annotate the data with this tagset because it is less biased towards a specific domain or set of semantic categories than, for example, a named-entity tagger. Using the same tagger as before we also annotated the text with a named-entity tagger trained on the BBN Wall Street Journal (WSJ) Entity Corpus which defines 105 categories for entities, nominal concepts, and numerical types.[11] See Figure 2 for a sample sentence annotated with these tags.

Next, we parsed all sentences with the dependency parser of Attardi et al. (2007).[12] We chose this parser because it is fast and it performed very well in the domain adaptation shared task of CoNLL 2007. Finally, we extracted semantic propositions using

9 You can request the corpus by email at `research-data-requests@yahoo-inc.com`. More information about this corpus can be found at: `http://www.yr-bcn.es/MannerYahooAnswers`.

10 `http://sourceforge.net/projects/supersensetag`.

11 LDC catalog number LDC2005T33.

12 `http://sourceforge.net/projects/desr`.

the SwiRL semantic parser of Surdeanu et al. (2007).[13] SwiRL starts by syntactically analyzing the text using a constituent-based full parser (Charniak 2000) followed by a semantic layer, which extracts PropBank-style semantic roles for all verbal predicates in each sentence.

It is important to realize that the output of all mentioned processing steps is noisy and contains plenty of mistakes, because the data have huge variability in terms of quality, style, genres, domains, and so forth. In terms of processing speed, both the semantic tagger of Ciaramita and Altun and the Attardi et al. parser process 100+ sentences/second. The SwiRL system is significantly slower: On average, it parses less than two sentences per second. However, recent research showed that this latter task can be significantly sped up without loss of accuracy (Ciaramita et al. 2008).

We used 60% of the questions for training, 20% for development, and 20% for testing. Our ranking model was tuned strictly on the development set for feature selection (described later) and the λ parameter of the translation models. The candidate answer set for a given question is composed of one positive example, that is, its corresponding best answer, and as negative examples all the other answers retrieved in the top N by the retrieval component.

4. Experiments

We used several measures to evaluate our models. Recall that we are using an initial retrieval engine to select a pool of N answer candidates (Figure 1), which are then re-ranked. This couples the performance of the initial retrieval engine and the re-rankers. We tried to de-couple them in our performance measures, as follows. We note that if the initial retrieval engine does not rank the correct answer in the pool of top N results, it is impossible for any re-ranker to do well. We therefore follow the approach of Ko et al. (2007) and define performance measures only with respect to the subset of pools which contain the correct answer for a given N.

This complicates slightly the typical notions of recall and precision. Let us call $\mathbf{Q}$ the set of all queries in the collection and $\mathbf{Q}^N$ the subset of queries for which the retrieved answer pool of size N contains the correct answer. We will then use the following performance measure definitions:

Retrieval Recall@N: The usual recall definition: $\frac{|\mathbf{Q}^N|}{|\mathbf{Q}|}$. This is equal for all re-rankers.

Re-ranking Precision@1: Average Precision@1 over the $\mathbf{Q}^N$ set, where the Precision@1 of a query is defined as 1 if the correct answer is re-ranked into the first position, 0 otherwise.

Re-ranking MRR: MRR over the $\mathbf{Q}^N$ set, where the reciprocal rank is the inverse of the rank of the correct answer.

Note that as N gets larger, $\mathbf{Q}^N$ grows in size, increasing the Retrieval Recall@N but also increasing the difficulty of the task for the re-ranker, and therefore decreasing Re-ranking Precision@1 and Re-ranking MRR.

During training of the FMIX re-ranker, the presentation of the training instances is randomized, which defines a randomized training protocol producing different models with each permutation of the data. We exploit this property to estimate the variance on the experimental results by reporting the average performance of 10 different models, together with an estimate of the standard deviation.

13 http://swirl-parser.sourceforge.net.

Table 3
Re-ranking evaluation for Perceptron and SVM-rank. Improvement indicates relative improvement over the baseline.

	$N = 15$	$N = 25$	$N = 50$	$N = 100$
Retrieval Recall@N	29.04%	32.81%	38.09%	43.42%
Re-ranking Precision@1				
Baseline	41.48	36.74	31.66	27.75
FMIX (Perceptron)	**49.87**$\pm$0.03	**44.48**$\pm$0.03	**38.53**$\pm$0.11	**33.72**$\pm$0.05
FMIX (SVM-rank)	49.48	44.10	38.18	33.52
Improvement (Perceptron)	**+20.22%**	**+21.06%**	**+21.69%**	**+21.51%**
Improvement (SVM-rank)	+19.28%	+20.03%	+20.59%	+20.79%
Re-ranking MRR				
Baseline	56.12	50.31	43.74	38.53
FMIX (Perceptron)	**64.16**$\pm$0.01	**58.20**$\pm$0.01	**51.19**$\pm$0.07	**45.29**$\pm$0.05
FMIX (SVM-rank)	63.81	57.89	50.93	45.12
Improvement (Perceptron)	**+14.32%**	**+15.68%**	**+17.03%**	**+17.54%**
Improvement (SVM-rank)	+13.70%	+15.06%	+16.43%	+17.10%

The initial retrieval engine used to select the pool of candidate answers is the BM25 score as described earlier. This is also our baseline re-ranker. We will compare this to the FMIX re-ranker using all features or using subsets of features.

4.1 Overall Results

Table 3 and Figure 4 show the results obtained using FMIX and the baseline for increasing values of *N*. We report results for Perceptron and SVM-rank using the optimal feature set for each (we discuss feature selection in the next sub-section).

Looking at the first column in Table 3 we see that a good bag-of-words representation alone (BM25 in this case) can achieve 41.5% Precision@1 (for the 29.0% of queries for

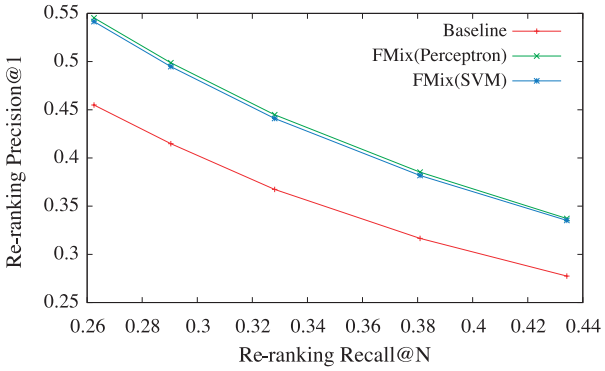

Figure 4
Re-ranking evaluation; precision-recall curve.

which the retrieval engine can find an answer in the top $N = 15$ results). These baseline results are interesting because they indicate that the problem is not hopelessly hard, but it is far from trivial. In principle, we see much room for improvement over bag-of-words methods. Indeed, the FMIX re-ranker greatly improves over the baseline. For example, the FMIX approach using Perceptron yields a Precision@1 of 49.9%, a 20.2% relative increase.

Setting N to a higher value we see recall increase at the expense of precision. Because recall depends only on the retrieval engine and not on the re-ranker, what we are interested in is the relative performance of our re-rankers for increasing numbers of N. For example, setting $N = 100$ we observe that the BM25 re-ranker baseline obtains 27.7% Precision@1 (for the 43.4% of queries for which the best answer is found in the top $N = 100$). For this same subset, the FMIX re-ranker using Perceptron obtains 33.7% Precision@1, a 21.5% relative improvement over the baseline model.

The FMIX system yields a consistent and significant improvement for all values of N, regardless of the type of learning algorithm used. As expected, as N grows the precision of both re-rankers decreases, but the relative improvement holds or increases. This can be seen most clearly in Figure 4 where re-ranking Precision and MRR are plotted against retrieval Recall. Recalling that the FMIX model was trained only once, using pools of $N = 15$, we can note that the training framework is stable at increasing sizes of N.

Table 3 and Figure 4 show that the two FMIX variants (Perceptron and SVM-rank) yield scores that are close (e.g., Precision@1 scores are within 0.5% of each other). We hypothesize that the small difference between the two different learning models is caused by our greedy tuning procedures (described in the next section), which converge to slightly different solutions due to the different learning algorithms. Most importantly, the fact that we obtain analogous results with two different learning models underscores the robustness of our approach and of our feature set.

These overall results provide strong evidence that: (a) readily available and scalable NLP technology can be used to improve lexical matching and translation models for retrieval and QA tasks, (b) we can use publicly available online QA collections to investigate features for answer ranking without the need for costly human evaluation, and (c) we can exploit large and noisy on-line QA collections to improve the accuracy of answer ranking systems. In the remainder of this section we analyze the performance of the different features.

4.2 Contribution of Feature Groups

In order to gain some insights about the effectiveness of the different features groups, we carried out a greedy feature selection procedure. We implemented similar processes for Perceptron and SVM-rank, to guarantee that our conclusions are not biased by a particular learning model.

4.2.1 Perceptron. We initialized the feature selection process with a single feature that replicates the baseline model (BM25 applied to the bag-of-words [W] representation). Then the algorithm incrementally adds to the feature set the single feature that provides the highest MRR improvement in the development partition. The process stops when no features yield any improvement. Note that this is only a heuristic process, and needs to be interpreted with care. For example, if two features were extremely correlated, the algorithm would choose one at random and discard the other. Therefore, if a feature is

missing from the selection process it means that it is either useless, or strongly correlated with other features in the list.

Table 4 summarizes the outcome of this feature selection process. Where applicable, we show within parentheses the text representation for the corresponding feature: W for words, N for *n*-grams, D for syntactic dependencies, and R for semantic roles. We use subscripts to indicate if the corresponding representation is fully lexicalized (no subscript), or its elements are replaced by WordNet supersenses (WNSS) or named-entity tags (WSJ). Where applicable, we use the l superscript to indicate if the corresponding structures are labeled. No superscript indicates unlabeled structures. For example, D_{WNSS} stands for unlabeled syntactic dependencies where the participating tokens are replaced by their WordNet supersense; R^{l}_{WSJ} stands for semantic tuples of predicates and labeled arguments with the words replaced with the corresponding WSJ named-entity tags.

The table shows that, although the features selected span all the four feature groups introduced, the lion's share is taken by the translation features (FG2): 75% of the MRR improvement is achieved by these features. The frequency/density features (FG3) are responsible for approximately 16% of the improvement. The rest is due to the query-log correlation features (FG4). This indicates that, even though translation models are the most useful, it is worth exploring approaches that combine several strategies for answer ranking.

As we noted before, many features may be missing from this list simply because they are strongly correlated with others. For example most similarity features (FG1) are correlated with BM25(W); for this reason the selection process does not choose a FG1 feature until iteration 9. On the other hand, some features do not provide a useful signal

Table 4
Summary of the model selection process using Perceptron.

Iteration	Feature Set	Group	MRR	P@1 (%)
0	BM25(W)	FG1	56.09	41.14
1	+ translation(R)	FG2	61.18	46.33
2	+ translation(N)	FG2	62.49	47.97
3	+ overall match(D_{WNSS})	FG3	63.07	48.93
4	+ translation(W)	FG2	63.27	49.12
5	+ query-log avg(PMI)	FG4	63.57	49.56
6	+ overall match(W)	FG3	63.72	49.74
7	+ overall match(W), normalized by Q size	FG3	63.82	49.89
8	+ same word sequence, normalized by Q size	FG3	63.90	49.94
9	+ BM25(N)	FG1	63.98	50.00
10	+ informativeness: verb count	FG3	64.16	49.97
11	+ query-log max(PMI)	FG4	64.37	50.28
12	+ same sentence match(W)	FG3	64.42	50.40
13	+ overall match(N_{WSJ})	FG3	64.49	50.51
14	+ query-log max(χ^2)	FG4	64.56	50.59
15	+ same word sequence	FG3	64.66	50.72
16	+ BM25(R_{WSJ})	FG1	64.68	50.78
17	+ translation(R^{l}_{WSJ})	FG2	64.71	50.75
18	+ answer span, normalized by A size	FG3	64.76	50.80
19	+ query-log top10(χ^2)	FG4	64.89	51.06
20	+ tree kernel(D_{WSJ})	FG3	64.93	51.07
21	+ translation(R_{WNSS})	FG2	64.95	51.16

at all. A notable example in this class is the Web-based CCP feature, which was designed originally for factoid answer validation and does not adapt well to our problem. To test this, we learned a model with BM25 and the Web-based CCP feature only, and this model did not improve over the baseline model at all. We hypothesize that because the length of non-factoid answers is typically significantly larger than in the factoid QA task, we have to discard a large part of the query when computing $hits(Q + A)$ to reach non-zero counts. This means that the final hit counts, hence the CCP value, are generally uncorrelated with the original (Q,A) tuple.

One interesting observation is that two out of the first three features chosen by our model selection process use information from the NLP processors. The first feature selected is the translation probability computed between the R representation (unlabeled semantic roles) of the question and the answer. This feature alone accounts for 57% of the measured MRR improvement. This is noteworthy: Semantic roles have been shown to improve factoid QA, but to the best of our knowledge this is the first result demonstrating that semantic roles can improve ad hoc retrieval (on a large set of non-factoid open-domain questions). We also find noteworthy that the third feature chosen measures the number of unlabeled syntactic dependencies with words replaced by their WNSS labels that are matched in the answer. Overall, the features that use the output of NL processors account for 68% of the improvement produced by our model over the IR baseline. These results provide empirical evidence that natural language analysis (e.g., coarse word sense disambiguation, syntactic parsing, and semantic role labeling) has a positive contribution to non-factoid QA, even in broad-coverage noisy settings based on Web data. To our knowledge, this had not been shown before.

Finally, we note that tree kernels provide minimal improvement: A tree kernel feature is selected only in iteration 20 and the MRR improvement is only 0.04 points. One conjecture is that, due to the sparsity and the noise of the data, matching trees of depth higher than 2 is highly uncommon. Hence matching immediate dependencies is a valid approximation of kernels in this setup. Another possible explanation is that because the syntactic trees produced by the parser contain several mistakes, the tree kernel, which considers matches between an exponential number of candidate subtrees, might be particularly unreliable on noisy data.

4.2.2 SVM-rank. For SVM-rank we employed a tuning procedure similar to the one used for the Perceptron that implements both feature selection and tuning of the regularizer parameter C. We started with the baseline feature alone and greedily added one feature at a time. In each iteration we added the feature that provided the best improvement. The procedure continues to evaluate all available features, until no improvement is observed. For this step we set the regularizer parameter to 1.0, a value which provided a good tradeoff between accuracy and speed as evaluated in an initial experiment. The selection procedure generated 12 additional features. At this point, using only the selected features, we fine-tuned the regularization parameter C across a wide spectrum of possible values. This can be useful because in SVM-rank the interpretation of C is slightly different than in standard SVM, specifically $C_{svm} = C_{rank}/m$, where m is the number of queries, or questions in our case. Therefore, an optimal value can depend crucially on the target data. The final value selected by this search procedure was equal to 290, although performance is relatively stable with values between 1 and 100,000. As a final optimization step, we continued the feature selection routine, starting from the 13 features already chosen and $C = 290$. This last step selected six additional features. A further attempt at fine-tuning the C parameter did not provide any improvements.

This process is summarized in Table 5, using the same notations as Table 4. Although the features selected by SVM-rank are slightly different than the ones chosen by the Perceptron, the conclusions drawn are the same as before: Features generated by NL processors provide a significant boost on top of the IR model. Similarly to the Perceptron, the first feature chosen by the selection procedure is a translation probability computed over semantic role dependencies (labeled, unlike the Perceptron, which prefers unlabeled dependencies). This feature alone accounts for 33.3% of the measured MRR improvement. This further enforces our observation that semantic roles improve retrieval performance for complex tasks such as our non-factoid QA exercise. All in all, 13 out of the 18 selected features, responsible for 70% of the total MRR improvement, use information from the NL processors.

4.3 Contribution of Natural Language Structures

One of the conclusions of the previous analysis is that features based on natural language processing are important for the problem of QA. This observation deserves a more detailed analysis. Table 6 shows the performance of our first three feature groups when they are applied to each of the content representations and incremental combinations of representations. In this table, for simplicity we merge features from labeled and unlabeled representations. For example, **R** indicates that features are extracted from both labeled (R^l) and unlabeled (R) semantic role representations. The g subscript indicates that the lexical terms in the corresponding representation are separately generalized to WNSS and WSJ labels. For example, $\mathbf{D}_g$ merges features generated from D_{WNSS},

Table 5
Summary of the model selection process using SVM-rank.

Iteration	Feature Set	Group	MRR	P@1 (%)
0	BM25(W)	FG1	56.09	41.12
1	+ translation(R^l)	FG2	59.02	43.99
2	+ answer span	FG3	60.31	45.05
3	+ translation(W)	FG2	61.16	46.13
4	+ translation(R)	FG2	61.65	46.77
5	+ overall match(D)	FG3	62.85	48.57
6	+ translation(R^l_{WSJ})	FG2	63.05	48.78
7	+ translation(N_{WSJ})	FG2	63.23	48.88
8	+ translation(D^l_{WSJ})	FG2	63.47	49.21
9	+ query-log max(χ^2)	FG4	63.64	49.35
10	+ translation(D)	FG2	63.77	49.53
11	+ translation(N)	FG2	63.85	49.66
12	+ overall match(N_{WSJ})	FG3	64.03	49.93
	+ C fine tuning		64.49	50.43
13	+ BM25(D_{WSJ})	FG1	64.49	50.43
14	+ BM25(N)	FG1	64.74	50.71
15	+ $tf \cdot idf$(D_{WNSS})	FG1	64.74	50.60
16	+ answer span in nouns	FG3	64.74	50.60
17	+ $tf \cdot idf$(R^l)	FG1	64.84	50.89
18	+ translation(D^l)	FG2	64.88	50.91

Table 6
Contribution of natural language structures in each feature group. Scores are MRR changes of the Perceptron on the development set over the baseline model (FG1 with W), for $N = 15$. The best scores for each feature group (i.e., column in the table), are marked in **bold**.

	FG1	FG2	FG3
W	–	+4.18	−6.80
N	−13.97	+2.49	−13.63
$\mathbf{N}_g$	−18.65	+3.63	−15.57
D	−15.15	+1.48	−15.39
$\mathbf{D}_g$	−19.31	+3.41	−18.18
R	−27.61	+0.33	−27.82
$\mathbf{R}_g$	−28.29	+3.46	−26.74
$\mathbf{W} + \mathbf{N}$	+1.46	+5.20	−4.36
$\mathbf{W} + \mathbf{N} + \mathbf{N}_g$	+1.51	+5.33	−4.31
$\mathbf{W} + \mathbf{N} + \mathbf{N}_g + \mathbf{D}$	+1.56	+5.78	−4.31
$\mathbf{W} + \mathbf{N} + \mathbf{N}_g + \mathbf{D} + \mathbf{D}_g$	+1.56	+5.85	**−4.21**
$\mathbf{W} + \mathbf{N} + \mathbf{N}_g + \mathbf{D} + \mathbf{D}_g + \mathbf{R}$	+1.58	+6.12	−4.28
$\mathbf{W} + \mathbf{N} + \mathbf{N}_g + \mathbf{D} + \mathbf{D}_g + \mathbf{R} + \mathbf{R}_g$	**+1.65**	**+6.29**	−4.28

D_{WSJ}, D^l_{WNSS}, and D^l_{WSJ}. For each cell in the table, we use only the features from the corresponding feature group and representation to avoid the correlation with features from other groups. We generate each best model using the same feature selection process described above.

The top part of the table indicates that all individual representations perform worse than the bag-of-words representation (W) in every feature group. The differences range from less than one MRR point (e.g., FG2[$\mathbf{R}_g$] versus FG2[**W**]), to over 28 MRR points (e.g., FG1[$\mathbf{R}_g$] versus FG1[**W**]). Such a large difference is justified by the fact that for feature groups FG1 and FG3 we compute feature values using only the corresponding structures (e.g., only semantic roles), which could be very sparse. For example, there are questions in our corpus where our SRL system does not detect any semantic proposition. Because translation models merge all structured representations with the bag-of-word representation, they do not suffer from this sparsity problem. Furthermore, on their own, FG3 features are significantly less powerful than FG1 or FG2 features. This explains why models using FG3 features fail to improve over the baseline. Regardless of these differences, the analysis indicates that in our noisy setting *the bag-of-words representation outperforms any individual structured representation*.

However, the bottom part of the table tells a more interesting story: The second part of our analysis indicates that structured representations provide *complementary* information to the bag-of-words representation. Even the combination of bag of words with the simplest *n*-gram structures ($\mathbf{W} + \mathbf{N}$) always outperforms the bag-of-words representation alone. But the best results are always obtained when the combination includes more natural language structures. The improvements are relatively small, but remarkable (e.g., see FG2) if we take into account the significant scale and settings of the evaluation. The improvements yielded by natural language structures are statistically significant for all feature groups. This observation correlates well with the analysis shown in Tables 4 and 5, which shows that features using semantic (R) and syntactic (D) representations contribute the most *on top* of the IR model (BM25(W)).

5. Error Analysis and Discussion

Similar to most re-ranking systems, our system improves the answer quality for some questions while decreasing it for others. Table 7 lists the percentage of questions from our test set that are improved (i.e., the correct answer is ranked higher after re-ranking), worsened (i.e., the correct answer is ranked lower), and unchanged (i.e., the position of the correct answer does not change after re-ranking). The table indicates that, regardless of the number of candidate answers for re-ranking (N), the number of improved questions is approximately twice the number of worsened questions. This explains the consistent improvements in P@1 and MRR measured for various values of N. As N increases, the number of questions that are improved also grows, which is an expected consequence of having more candidate answers to re-rank. However, the percentage of improved questions grows at a slightly lower rate than the percentage of worsened questions. This indicates that choosing the ideal number of candidate answers to re-rank requires a trade-off: On the one hand, having more candidate answers increases the probability of capturing the correct answer in the set; on the other hand, it also increases the probability of choosing an incorrect answer due to the larger number of additional candidates. For our problem, it seems that re-ranking using values of N much larger than 100 would not yield significant benefits over smaller values of N. This analysis is consistent with the experiments reported in Table 3 where we did not measure significant growth in P@1 or MRR for N larger than 50.

Although Table 7 gives the big picture of the behavior of our system, it is important to look at actual questions that are improved or worsened by the re-ranking model in order to understand the strengths and weaknesses of our system. Table 8 lists some representative questions where the re-ranking model brings the correct answer to the top position. For every question we list: (a) the correct answer and its position as given by the baseline IR Model ("Baseline") and the re-ranking model ("Re-ranking"); and (b) the answer that was ranked by the baseline model in the first position and its position after re-ranking.

Generally, Table 8 indicates that our model performs considerably better than the bag-of-words IR model. For example, we boost the rank of answers that share structures with the question: for example, the *cook* → *grouse* syntactico-semantic dependency for the second sample question or *make* → *call* and *see* → *number* for the third example. Modeling structures is important especially for questions with minimal context, that is, short length and common terms, like the third sample question. Due to the structure-based translation models and/or the generalizations to supersenses or named-entity labels, our model can match structures even when they are not identical. For example,

Table 7
Percentage of questions in the test set that are improved/worsened/unchanged after re-ranking. This experiment used the Perceptron model.

	Better (%)	Worse (%)	Unchanged (%)
$N = 10$	33.98	16.81	49.21
$N = 15$	36.76	18.56	44.68
$N = 25$	39.64	20.68	39.68
$N = 50$	42.95	23.30	33.75
$N = 100$	45.18	25.28	29.54

Table 8
Examples of questions improved by our re-ranking model. URLs were replaced with <URL> in answer texts. Some non-relevant text was replaced with <. . .> to save space. The remaining text maintains the original capitalization and spelling. Non-stop question terms are emphasized in the answers.

How would you rank the top 5 NFL teams? Do your rankings depend on the outcome of the Colts Vs. Pats Game?

Baseline	Re-ranking	Correct?	Answer Text
2	1	yes	Ok. How can you think the Chargers are better than the *Colts*, Ravens, and the Broncos? As for the *Rankings*, this is what I got: 1. *Colts* (they actually beat a good *team*) 2. Bears (for now, they are going to lose to the Giants) 3. Patriots (the winner of Cotls vs Patriots should be the next #1) 4. Broncos (They lost on a field goal) 5. Ravens (all about the D)
1	2	no	Basically in sport *rankings* you will look at the how they perform their wins against their loss or their stats. *Ranking* is the process of positioning individuals, groups or businesses on a ordinal scale in relation to others. A list arranged in this way is said to be in *rank* order. Some examples: In many sports, individuals or *teams* are given *rankings*, generally by the sport's governing body. In football (soccer), national *teams* are ranked in the FIFA World *Rankings*. In snooker, players are ranked using the Snooker world *rankings*. In ice hockey, national *teams* are ranked in the IIHF World *Ranking*. In golf, the top male golfers are *ranked* using the Official World Golf *Rankings*

how can i cook grouse quick with normal household spices w/o going out to buy stuff?

Baseline	Re-ranking	Correct?	Answer Text
10	1	yes	I've never *cooked* a *grouse*, but poultry is poultry... You could salt it and pepper it, put some flour on a plate and roll it in the flour to coat it lightly, then heat a few tablespoons of olive oil in a skillet and pan-fry it. (If you have no olive oil, use a little vegetable oil plus a pat of butter – the oil is to keep the butter from burning.) Squeeze a few drops of lemon juice over it if you want. Or: Skip the flour. Salt and pepper the grouse. Pan-fry it in a little olive oil. When it's looking close to ready, pour in a little white wine which will sizzle and reduce down to a sauce. If you've got some fresh or dried herbs (rosemary, thyme, parsley) you could sprinkle a pinch of that in.
1	2	no	Well, a *grouse* is a prey animal. If there was a decline in the population of *grouse*, then the animals that usually prey on the *grouse* - coyotes, owls, etc - would probably start eating other prey animals, like the pheasants and squirrels.

how did I do for make a call and that the other dont see my number?

Baseline	Re-ranking	Correct?	Answer Text
2	1	yes	to *make* a *call* so that the other person cant *see* the *number*... dial *67 and wait for the three beeps.. then dial the *number*
1	2	no	Oneday out of the blue *call* her. If u dont have her *number*, when u see her ask her if she wanted to go out oneday then get her *number*. When u talk on the phone get to know her. But dont ask her out too soon because she may not feel the same way. After a couple of days or weeks taking to her let her know how u felt about her since the first time u met her.

Table 8
(*continued*)

how can i find a veterinary college with dorms?			
Baseline	Re-ranking	Correct?	Answer Text
14	1	yes	<. . .> I would say not to look for a specific school of *veterinarian* medicine but rather find a creditable University that offers a degree such as Pre-Vet. Then from there you can attend graduate school to finish up to become a doctor in that field. Most major universities will have this degree along with *dorms*. In my sources you can see that this is just one of many major universities that offer Pre-vet medicine.
1	7	no	Hi there... here's an instructional video by Cornell University Feline Health Center - *College* of *Veterinary* Medicine on how to pill cats: <URL>
how to handle commission splits with partners in Real estate?			
Baseline	Re-ranking	Correct?	Answer Text
5	1	yes	My company *splits* the *commissions* all evenly. However many various agents/brokers are involved (or even think they are involved), it gets *split* further. Keeps everyone happy. No one complains that someone gets "more".
1	3	no	You will find information regarding obtaining a *real estate* license in Oklahoma at the Oklahoma *Real Estate Commission*'s website (<URL>) Good luck!

for the fourth question, *find* → *college* can be matched to *look* → *school* if the structures are generalized to WordNet supersenses. Translation models are crucial to fetching answers rich in terms related to question concepts. For example, for the first question, our model boosts the position of the correct answer due to the large numbers of concepts that are related to *NFL*, *Colts*, and *Pats*: *Ravens*, *Broncos*, *Patriots*, and so forth. In the second example, our model ranks on the first position the answer containing many concepts related to *cook*: *salt*, *pepper*, *flour*, *tablespoons*, *oil*, *skillet*, and so on. In the last example, our model is capable of associating the bigram *real estate* to *agent* and *broker*. Without these associations many answers are lost to false positives provided by the bag-of-words similarity models. For example, in the first and last examples in the table, the answers selected by the baseline model contain more matches of the questions terms than the correct answers extracted by our model.

All in all, this analysis proves that non-factoid QA is a complex problem where many phenomena must be addressed. The key for success does not seem to be a unique model, but rather a combination of approaches each capable of addressing different facets of the problem. Our model makes a step forward towards this goal, mainly through concept expansion and the exploration of syntactico-semantic structures. Nevertheless, our model is not perfect. To understand where FMIX fails we performed a manual error analysis on 50 questions where FMIX performs worse than the IR baseline and we identified seven error classes. Table 9 lists the distribution of these error classes and Table 10 lists sample questions and answers from each class. Note that the percentage values listed in Table 9 sum up to more than 100% because the error classes are not exclusive. We now detail each of these error classes.

Table 9
Distribution of error classes in questions where FMIX (Perceptron) performs worse.

COMPLEX INFERENCE	38%
ELLIPSIS	36%
ALSO GOOD	18%
REDIRECTION	10%
ANSWER QUALITY	4%
SPELLING	2%
CLARIFICATION	2%

COMPLEX INFERENCE: This is the most common class of errors (38%). Questions in this class could theoretically be answered by an automated system but such a system would require complex reasoning mechanisms, large amounts of world knowledge, and discourse understanding. For example, to answer the first question in Table 10, a system would have to understand that *confronting* or *being supportive* are forms of *dealing with* a person. To answer the second question, the system would have to know that creating a CD *at what resolution you need* supersedes *making a low resolution CD*. Our approach captures some simple inference rules through translation models but fails to understand complex implications such as these.

ELLIPSIS: This class of errors is not necessarily a fault of our approach but is rather caused by the problem setting. Because in a social QA site each answer responds to a specific question, discourse ellipsis (i.e., omitting the context set by the question in the answer text) is common. This makes some answers (e.g., the third answer in Table 10) ambiguous, hence hard to retrieve automatically. This affects 36% of the questions analyzed.

ALSO GOOD: It is a common phenomenon in Yahoo! Answers that a question is asked several times by different users, possibly in a slightly different formulation. To enable our large scale automatic evaluation, we considered an answer as correct only if it was chosen as the "best answer" for the corresponding question. So in our setting, "best answers" from equivalent questions are marked as incorrect. This causes 18% of the "errors" of the re-ranking model. One example is the fourth question in Table 10, where the answer selected by our re-ranking model is obviously also correct. It is important to note that at testing time we do not have access to the questions that generated the candidate answers for the current test question, that is, the system does not know which questions are answered by the answers in the ALSO GOOD section of Table 10. So the answers in the ALSO GOOD category are not selected based on the similarity of the corresponding queries, but rather, based on better semantic matching between test question and candidate answer.

REDIRECTION: Some answers (10% of the questions analyzed) do not directly answer a question but rather redirect the user to relevant URLs (see the fifth question in Table 10). Because we do not extract the text behind URLs in the answer content, such questions are virtually impossible to answer using our approach.

ANSWER QUALITY: For a small number of the questions analyzed (4%) the choice of "best answer" is dubious (see the sixth example in Table 10). This is to be expected in a social QA site, where the selection of best answers is not guaranteed to be optimal. Nevertheless, the relatively small number of such cases is unlikely to influence the quality of the evaluation.

Table 10
Examples of questions in each error class. The corresponding error class is listed on the left side of the question text. We list the answer ranked at the top position by FMIX only where relevant (e.g., the ALSO GOOD category). URLs were replaced with <URL> in answer texts. Some non-relevant text was replaced with <. . .> to save space. The remaining text maintains the original capitalization and spelling.

COMPLEX INFERENCE	how to deal with a person in denial with M.P.D.?		
Baseline	Re-ranking	Correct?	Answer Text
1	6	yes	First, i would find out if MPD has been diagnosed by a professional. In current terminology, MPD is considered a part of Dissociative Personality Disorder. In any case, it would be up to the professionals to help this person because you could cause further problems by confronting this person with what you think the problem is. If this person is a family member, you could ask for a consultation with the psychiatric professional who is treating him/her. Please, please, just do you best to be supportive without being confrontational since that might make things even worse for that person.
COMPLEX INFERENCE	How do I make a low resolution CD of 100 phtos that were shot at 8 megapixels?		
Baseline	Re-ranking	Correct?	Answer Text
1	4	yes	you can use picasa2 from google: <URL> is free. In picasa you can do "GiftCD" create a CD with you picture at what resolution you need (including original size)
ELLIPSIS	How do mineral ions affect the biology and survival of a pond organism?		
Baseline	Re-ranking	Correct?	Answer Text
1	3	yes	Some mineral ions are fertilizer and will increase vegetative growth while others are poisons.
ALSO GOOD	How to learn the British accent?		
Baseline	Re-ranking	Correct?	Answer Text
4	5	yes	Get a dictionary where there is a pronunciation guide which gives the pronunciation in British English. Watch british movies and imitate what you can. Then just practice, practice practice. But before you go about learning accents, slangs or dialects, make sure you brush up on your basic grammar. <. . .>
3	1	no	You can do one of two things: first, go to a local bookstore, like Barnes and Noble. They sell cd's with different accents from around the world, accompanied by a book that phonetically spells the words. This is designed for actors/actresses who need to learn different accents. Also, go rent a bunch of british movies, or watch british television. Continually pause and repeat common phrases and words.

Table 10
(*continued*)

REDIRECTION	How can I build an easy lean-to shed out of scrap wood and skids?		
Baseline	Re-ranking	Correct?	Answer Text
6	15	yes	the pallet shed... <URL> building a wood shed from pallets... <URL> good ideas from those who've been there...<URL> pics. of the shed... <URL> nice pics. <URL> taking pallets apart... and other tips... <URL> <. . .>
ANSWER QUALITY	How do make a Naruto AMV? Can you show me how? I need the website or program and the exact directions.?		
Baseline	Re-ranking	Correct?	Answer Text
2	94	yes	i'm not an expert. but i sure do like Naruto. i'll wait for answers too
SPELLING	how does aliquid expansion boiler thrrmosstat work?		
Baseline	Re-ranking	Correct?	Answer Text
2	4	yes	the liquid expands inside the thermostat when the liquid reaches the shutoff temp or pressure it will shut off the boiler preventing boiler explosions
CLARIFICATION	how could you combine your styles and personalities effectively to produce the best paper?		
Baseline	Re-ranking	Correct?	Answer Text
29	1	yes	Your question is not clear. Are you asking about writing styles? it also depends on what kind of paper you are writing? Your question cannot be answered without more info.

SPELLING: Two percent (2%) of the error cases analyzed are caused by spelling errors (e.g., the seventh example in Table 10). Because these errors are relatively infrequent, they are not captured by our translation models, and our current system does not include any other form of spelling correction.

CLARIFICATION: Another 2% of the questions inspected manually had answers that pointed to errors or ambiguities in the question text rather than responding to the given question (see the last example in Table 10). These answers are essentially correct but they require different techniques to be extracted: Our assumption is that questions are always correct and sufficient for answer extraction.

6. Related Work

There is a considerable amount of previous work in several related areas. First, we will discuss related work with respect to the features and models used in this research; most of this work is to be found in the factoid QA community, where the most sophisticated

QA selection and re-ranking algorithms have been developed. We then review existing work in non-factoid QA; we will see that in this area there is much less work, and the emphasis has been so far in query re-writing and scalability using relatively simple features and models. Finally we will discuss related work in the area of community-built (**social**) QA sites. Although we do not exploit the social aspect of our QA collection, this is complementary to our work and would be a natural extension. Table 11 summarizes aspects of the different approaches discussed in this section, highlighting the differences and similarities with our current work.

Our work borrows ideas from many of the papers mentioned in this section, especially for feature development; indeed our work includes matching features as well as translation and retrieval models, and operates at the lexical level, the parse tree

Table 11
Comparison of some of the characteristics of the related work cited. **Task**: Document Retrieval (DRet), Answer Extraction (Ex) or Answer Re-ranking or Selection (Sel). **Queries**: factoid (Fact) or non-factoid (NonFact). **Features**: lexical (L), *n*-grams (Ngr), collocations (Coll), paraphrases (Para), POS, syntactic dependency tree (DT), syntactic constituent tree (CT), named entities (NE), WordNet Relations (WNR), WordNet supersenses (WNSS), semantic role labeling (SRL), causal relations (CR), query classes (QC), query-log co-ocurrences (QLCoOcc). **Models**: bag-of-words scoring (BOW), tree matching (TreeMatch), linear (LM), log-linear (LLM), statistical learning (kernel) (SL), probabilistic grammar (PG), statistical machine translation (SMT), query likelihood language model (QLLM). **Development** and **Evaluation**: data sizes used, expressed as number of queries/number of query–answer pairs (i.e., sum of all candidate answers per question). **Data**: type of source used for feature construction, training and/or evaluation. Question marks are place holders for information not available or not applicable in the corresponding work.

Publication	Task	Queries	Features	Models	Devel	Eval	Data
Agichtein et al. (2001)	DRet	NonFact	L, Ngr, Coll	BOW, LM	?/10K	50/100	WWW
Echihabi and Marcu (2003)	Sel	Fact, NonFact	L, Ngr, Coll, DT, NE, WN	SMT	4.6K/100K	1K/300K?	TREC, KM, WWW
Higashinaka and Isozaki (2008)	Sel	NonFact (Why)	L, WN, SRL, CR	SL	1K/500K	1K/500K	WHYQA
Punyakanok et al. (2004)	Sel	Fact	L, POS, DT, NE, QC	TreeMatch		?/400	TREC13
Riezler et al. (2007)	DRet	NonFact	L, Ngr, Para	SMT	10M/10M	60/1.2K	WWW, FAQ
Soricut and Brill (2006)	DRet, Sel, Ex	NonFact	L, Ngr, Coll	BOW, SMT	1M/?	100/?	WWW, FAQ
Verberne et al. (2010)	Sel	NonFact (Why)	CT, WN, Para	BOW, LLM	same as eval	186/28K	Webclopedia, Wikipedia
Wang et al. (2007)	Sel	Fact	L, POS, DT, NE, WNR, Hyb	LLM, PG	100/1.7K	200/1.7K	TREC13
Xue et al. (2008)	DRet, Sel	NonFact, Fact	L, Coll	SMT, QLLM	1M/1M	50/?	SocQA, TREC9
This work	Sel	NonFact (How)	L, Ngr, POS, DT, SRL, NE, WN, WNSS, Hyb, QLCoOcc	TreeMatch, BOW, SMT, SL	112K/1.6M	28K/up to 2.8M	SocQA, QLog

level, as well as the level of semantic roles, named entities, and lexical semantic classes. However, to the best of our knowledge no previous work in QA has evaluated the use of so many types of features concurrently, nor has it built so many combinations of these features at different levels. Furthermore, we employ unsupervised methods, generative methods, and supervised learning methods. This is made possible by the choice of the task and the data collection, another novelty of our work which should enable future research in complex linguistic features for QA and ranking.

Factoid QA. Within the statistical machine translation community there has been much research on the issue of automatically learning transformations (at the lexical, syntactical, and semantical level). Some of this work has been applied to automated QA systems, mostly for factoid questions. For example, Echihabi and Marcu (2003) presented a noisy-channel approach (IBM model 4) adapted for the task of QA. The features used included lexical and parse-tree elements as well as some named entities (such as dates). They use a dozen heuristic rules to heavily reduce the feature space and choose a single representation mode for each of the tokens in the queries (for example: *"terms overlapping with the question are preserved as surface text")* and learn language models on the resulting representation. We extend Echihabi and Marcu by considering deeper semantic representations (such as SRL and WNSS), but instead of using selection heuristics we learn models from each of the full representations (as well as from some hybrid representations) and then combine them using discriminant learning techniques.

Punyakanok, Roth, and Yih (2004) attempted a more comprehensive use of the parse tree information, computing a similarity score between question and answer parse trees (using a distance function based on approximate tree matching algorithms). This is an unsupervised approach, which is interesting especially when coupled with appropriate distances. Shen and Joshi (2005) extend this idea with a supervised learning approach, training dependency tree kernels to compute the similarity. In our work we also used this type of feature, although we show that, in our context, features based on dependency tree kernels are subsumed by simpler features that measure the overlap of binary dependencies. Another alternative is proposed by Cui et al. (2005), where significant words are aligned and similarity measures (based on mutual information of correlations) are then computed on the resulting dependency paths. Shen and Klakow (2006) extend this using a dynamic time warping algorithm to improve the alignment for approximate question phrase mapping, and learn a Maximum Entropy model to combine the obtained scores for re-ranking. Wang, Smith, and Mitamura (2007) propose to use a probabilistic quasi-synchronous grammar to learn the syntactic transformations between questions and answers. We extend the work of Cui et al. by considering paths within and across different representations beyond dependency trees, although we do not investigate the issue of alignment specifically—instead we use standard statistical translation models for this.

Non-factoid QA. The previous works dealt with the problem of **selection**, that is, finding the single sentence that correctly answers the question out of a set of candidate documents. A related problem in QA is that of **retrieval**: selecting potentially relevant documents or sentences prior to the selection phase. This problem is closer to general document retrieval and it is therefore easier to generalize to the non-factoid domain. Retrieval algorithms tend to be much simpler than selection algorithms, however, in part due to the need for speed, but also because there has been little previous evidence that complex algorithms or deeper linguistic analysis helps at this stage, especially in the context of non-factoid questions.

Previous work addressed the task by learning transformations between questions and answers and using them to improve retrieval. All these works use only lexical features. For example, Agichtein et al. (2001) learned lexical transformations (from the original question to a set of Web search queries, from "*what is a*" to *"the term"*, "*stands for*", etc.) which are likely to retrieve good candidate documents in commercial Web search engines; they applied this successfully to large-scale factoid and non-factoid QA tasks. Murdock and Croft (2005) study the problem of candidate sentence retrieval for QA and show that a lexical translation model can be exploited to improve factoid QA. Xue, Jeon, and Croft (2008) show that a linear interpolation of translation models and a query likelihood language model outperforms each individual model for a QA task that is independent of the question type. In the same space, Riezler et al. (2007) develop SMT-based query expansion methods and use them for retrieval from FAQ pages. In our work we did not address the issue of query expansion and re-writing directly: While our re-ranking approach is limited to the recall of the retrieval model, these methods of query transformation could be used in a complementary manner to improve the recall. Even more interesting would be to couple the two approaches in an efficient manner; this remains as future work.

There has also been some work in the problem of selection for non-factoid questions. Girju (2003) extracts non-factoid answers by searching for certain semantic structures (e.g., causation relations as answers to causation questions). We generalized this methodology (in the form of semantic roles) and evaluated it systematically. Soricut and Brill (2006) develop a statistical model by extracting (in an unsupervised manner) QA pairs from one million FAQs obtained from the Web. They show how different statistical models may be used for the problems of ranking, selection, and extraction of non-factoid QAs on the Web; due to the scale of their problem they only consider lexical *n*-grams and collocations, however. More recent work has showed that structured retrieval improves answer ranking for factoid questions: Bilotti et al. (2007) showed that matching predicate–argument frames constructed from the question and the expected answer types improves answer ranking. Cui et al. (2005) learned transformations of dependency paths from questions to answers to improve passage ranking. All these approaches use similarity models at their core because they require the matching of the lexical elements in the search structures, however. On the other hand, our approach allows the learning of full transformations from question structures to answer structures using translation models applied to different text representations.

The closest work to ours is that of Higashinaka and Isozaki (2008) and Verberne et al. (2010), both on *Why* questions. Higashinaka et al. consider a wide range of semantic features by exploiting WordNet and gazetteers, semantic role labeling, and extracted causal relations. Verberne et al. exploit syntactic information from constituent trees, WordNet synonymy sets and relatedness measures, and paraphrases. As in our models, both these works combine these features using discriminative learning techniques and apply the learned models to re-rank answers to non-factoid questions (*Why* type questions). Their features, however, are based on counting matches or events defined heuristically. We have extended this approach in several ways. First, we use a much larger feature set that includes correlation and transformation-based features and five different content representations. Second, we use generative (translation) models to learn transformation functions before they are combined by the discriminant learner. Finally, we carry out training and evaluation at a much larger scale.

Content from community-built question–answer sites can be retrieved by searching for similar questions already answered (Jeon, Croft, and Lee 2005) and ranked using meta-data information like answerer authority (Jeon et al. 2006; Agichtein et al. 2008).

Here we show that the answer text can be successfully used to improve answer ranking quality. Our method is complementary to the earlier approaches. It is likely that an optimal retrieval engine from social media would combine all three methodologies. Moreover, our approach might have applications outside of social media (e.g., for open-domain Web-based QA), because the ranking model built is based only on open-domain knowledge and the analysis of textual content.

7. Conclusions

In this work we describe an answer ranking system for non-factoid questions built using a large community-generated question–answer collection. We show that the best ranking performance is obtained when several strategies are combined into a single model. We obtain the best results when similarity models are aggregated with features that model question-to-answer transformations, frequency and density of content, and correlation of QA pairs with external collections. Although the features that model question-to-answer transformations provide the most benefits, we show that the combination is crucial for improvement. Further, we show that complex linguistic features, most notably semantic role dependencies and semantic labels derived from WordNet senses, yield a statistically significant performance increase on top of the traditional bag-of-words and *n*-gram representations. We obtain these results using only off-the-shelf NL processors that were not adapted in any way for our task. As a side effect, our experiments prove that we can effectively exploit large amounts of available Web data to do research on NLP for non-factoid QA systems, without any annotation or evaluation cost. This provides an excellent framework for large-scale experimentation with various models that otherwise might be hard to understand or evaluate.

As implications of our work, we expect the outcome of our investigation to help several applications, such as retrieval from social media and open-domain QA on the Web. On social media, for example, our system should be combined with a component that searches for similar questions already answered; the output of this ensemble can possibly be filtered further by a content-quality module that explores "social" features such as the authority of users, and so on. Although we do not experiment on Wikipedia or news sites in this work, one can view our data as a "worse-case scenario," given its ungrammaticality and annotation quality. It seems reasonable to expect that training our model on cleaner data (e.g., from Wikipedia or news), would yield even better results.

This work can be extended in several directions. First, answers that were not selected as best, but were marked as good by a minority of voters, could be incorporated in the training data, possibly introducing a graded notion of relevance. This would make the learning problem more interesting and would also provide valuable insights into the possible pitfalls of user-annotated data. It is not clear if more data, but of questionable quality, is beneficial. Another interesting problem concerns the adaptation of the re-ranking model trained on social media to collections from other genres and/or domains (news, blogs, etc.). To our knowledge, this domain adaptation problem for QA has not been investigated yet.

References

Agichtein, Eugene, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. 2008. Finding high-quality content in social media, with an application to community-based question answering. In *Proceedings of the Web Search and Data Mining Conference (WSDM)*, pages 183–194, Stanford, CA.

Agichtein, Eugene, Steve Lawrence, and Luis Gravano. 2001. Learning search engine specific query transformations for question answering. In *Proceedings of the World*

Wide Web Conference, pages 169–178, Hong Kong.

Attardi, Giuseppe, Felice Dell'Orletta, Maria Simi, Atanas Chanev, and Massimiliano Ciaramita. 2007. Multilingual dependency parsing and domain adaptation using DeSR. In *Proceedings of the Shared Task of the Conference on Computational Natural Language Learning (CoNLL)*, pages 1112–1118, Prague.

Berger, Adam, Rich Caruana, David Cohn, Dayne Freytag, and Vibhu Mittal. 2000. Bridging the lexical chasm: Statistical approaches to answer finding. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research & Development on Information Retrieval*, pages 192–199, Athens, Greece.

Bilotti, Matthew W., Paul Ogilvie, Jamie Callan, and Eric Nyberg. 2007. Structured retrieval for question answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research & Development on Information Retrieval*, pages 351–358, Amsterdam.

Brill, Eric, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. 2001. Data-intensive question answering. In *Proceedings of the Text REtrieval Conference (TREC)*, pages 393–400, Gaithersburg, MD, USA.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 132–139, Seattle, WA.

Ciaramita, Massimiliano and Yasemin Altun. 2006. Broad coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 594–602, Sidney.

Ciaramita, Massimiliano, Giuseppe Attardi, Felice Dell'Orletta, and Mihai Surdeanu. 2008. Desrl: A linear-time semantic role labeling system. In *Proceedings of the Shared Task of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*, pages 258–262, Manchester.

Ciaramita, Massimiliano and Mark Johnson. 2003. Supersense tagging of unknown nouns in WordNet. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 168–175, Sapporo.

Ciaramita, Massimiliano, Vanessa Murdock, and Vassilis Plachouras. 2008. Semantic associations for contextual advertising. *Journal of Electronic Commerce Research—Special Issue on Online Advertising and Sponsored Search*, 9(1):1–15.

Collins, Michael and Nigel Duffy. 2001. Convolution kernels for natural language. In *Proceedings of the Neural Information Processing Systems Conference (NIPS)*, pages 625–632, Vancouver, Canada.

Cui, Hang, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 400–407, Salvador.

Echihabi, Abdessamad and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 16–23, Sapporo.

Freund, Yoav and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37:277–296.

Girju, Roxana. 2003. Automatic detection of causal relations for question answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL), Workshop on Multilingual Summarization and Question Answering*, pages 76–83, Sapporo.

Harabagiu, Sanda, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2000. Falcon: Boosting knowledge for answer engines. In *Proceedings of the Text REtrieval Conference (TREC)*, pages 479–487, Gaithersburg, MD.

Higashinaka, Ryuichiro and Hideki Isozaki. 2008. Corpus-based question answering for why-questions. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP)*, pages 418–425, Hyderabad.

Jeon, Jiwoon, W. Bruce Croft, and Joon Hoo Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the ACM Conference on Information and Knowledge*

Management (CIKM), pages 84–90, Bremen.

Jeon, Jiwoon, W. Bruce Croft, Joon Hoo Lee, and Soyeon Park. 2006. A framework to predict the quality of answers with non-textual features. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 228–235, Seattle, WA.

Joachims, Thorsten. 2006. Training linear svms in linear time. In *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226, New York, NY.

Ko, Jeongwoo, Teruko Mitamura, and Eric Nyberg. 2007. Language-independent probabilistic answer ranking for question answering. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 784–791, Prague.

Li, Xin and Dan Roth. 2006. Learning question classifiers: The role of semantic information. *Natural Language Engineering*, 12:229–249.

Li, Yaoyong, Hugo Zaragoza, Ralf Herbrich, John Shawe-Taylor, and Jaz S. Kandola. 2002. The perceptron algorithm with uneven margins. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 379–386, Sidney.

Magnini, Bernardo, Matteo Negri, Roberto Prevete, and Hristo Tanev. 2002. Comparing statistical and content-based techniques for answer validation on the web. In *Proceedings of the VIII Convegno AI*IA*, Siena, Italy.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Moldovan, Dan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Richard Goodrum, Roxana Girju, and Vasile Rus. 1999. Lasso—a tool for surfing the answer net. In *Proceedings of the Text REtrieval Conference (TREC)*, pages 175–183, Gaithersburg, MD.

Moschitti, Alessandro, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 776–783, Prague.

Murdock, Vanessa and W. Bruce Croft. 2005. A translation model for sentence retrieval. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 684–691, Vancouver.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106

Punyakanok, Vasin, Dan Roth, and Wen-tau Yih. 2004. Mapping dependencies trees: An application to question answering. *Proceedings of AI&Math 2004*, pages 1–10, Fort Lauderdale, FL.

Riezler, Stefan, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 464–471, Prague.

Robertson, Stephen and Stephen G. Walker. 1997. On relevance weights with little relevance information. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 16–24, New York, NY.

Shen, Dan and Dietrich Klakow. 2006. Exploring correlation of dependency relation paths for answer extraction. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 889–896, Sydney.

Shen, Libin and Aravind K. Joshi. 2005. Ranking and reranking with perceptron. *Machine Learning. Special Issue on Learning in Speech and Language Technologies*, 60(1):73–96.

Soricut, Radu and Eric Brill. 2006. Automatic question answering using the Web: Beyond the factoid. *Journal of Information Retrieval—Special Issue on Web Information Retrieval*, 9(2):191–206.

Surdeanu, Mihai and Massimiliano Ciaramita. 2007. Robust information extraction with perceptrons. In *Proceedings of the NIST 2007 Automatic Content Extraction Workshop (ACE07)*, College Park, MD. Available at: `http://www.surdeanu.name/mihai/papers/ace07a.pdf`.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluis Marquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic

dependencies. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 159–177, Manchester.

Surdeanu, Mihai, Lluis Marquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29:105–151.

Tao, Tao and ChengXiang Zhai. 2007. An exploration of proximity measures in information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 259–302, Amsterdam.

Tsochantaridis, Ioannis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *ICML '04: Proceedings of the Twenty-First International Conference on Machine Learning*, pages 104–111, New York, NY.

Verberne, Suzan, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2010. What is not in the bag of words for why-qa? *Computational Linguistics*, 36(2):229–245.

Voorhees, Ellen M. 2001. Overview of the TREC-9 question answering track. In *Proceedings of the Text REtrieval Conference (TREC) TREC-9 Proceedings*, pages 1–15, Gaithersburg, MD.

Wang, Mengqiu, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 22–32, Prague.

Xue, Xiaobing, Jiwoon Jeon, and W. Bruce Croft. 2008. Retrieval models for question and answer archives. In *Proceedings of the Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 475–482, Singapore.