Learning to Rank Answers on Large Online QA Collections

Mihai Surdeanu, Massimiliano Ciaramita, Hugo Zaragoza
Barcelona Media Innovation Center, Yahoo! Research Barcelona
mihai.surdeanu@barcelonamedia.org, {massi,hugo}@yahoo-inc.com

Abstract

This work describes an answer ranking engine for non-factoid questions built using a large online community-generated question-answer collection (Yahoo! Answers). We show how such collections may be used to effectively set up large supervised learning experiments. Furthermore we investigate a wide range of feature types, some exploiting NLP processors, and demonstrate that using them in combination leads to considerable improvements in accuracy.

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.
Low Quality	*Q*: How to extract html tags from an html documents with c++? *A*: very carefully

Table 1: Sample content from Yahoo! Answers.

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)[1] or the Cross Language Evaluation Forum (CLEF) Workshop[2], 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.

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). 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 tends to be domain-specific, hence unsuitable for the learning of open-domain models.

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[3], 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. The data generated by these sites has significant advantages over other web resources: (a) it has a high growth rate and it is already abundant; (b) it covers a large number of topics, hence it offers a better

[1] http://trec.nist.gov

[2] http://www.clef-campaign.org

[3] http://answers.yahoo.com

Proceedings of ACL-08: HLT, pages 719–727,
Columbus, Ohio, USA, June 2008.

approximation of open-domain content; and (c) it is available for many languages. Community QA sites, similar to FAQs, provide large number of question-answer pairs. Nevertheless, this data has a significant drawback: it has high variance of quality, i.e., answers range from very informative to completely irrelevant or even abusive. Table 1 shows some examples of both high and low quality content.

In this paper we address the problem of answer ranking for non-factoid questions from social media content. Our research objectives focus on answering the following two questions:

1. *Is it possible to learn an answer ranking model for complex questions from such noisy data?* This is an interesting question because a positive answer indicates that a plethora of training data is readily available to QA researchers and system developers.
2. *Which features are most useful in this scenario?* Are similarity models as effective as models that learn question-to-answer transformations? Does syntactic and semantic information help? For generality, we focus only on textual features extracted from the answer text and we ignore all meta data information that is not generally available.

Notice that we concentrate on one component of a possible social-media QA system. In addition to answer ranking, a complete system would have to search for similar questions already answered (Jeon et al., 2005), and rank content quality using ”social” features such as the authority of users (Jeon et al., 2006; Agichtein et al., 2008). This is not the focus of our work: here we investigate the problem of learning an answer ranking model capable of dealing with complex questions, using a large number of, possible noisy, question-answer pairs. By focusing exclusively on textual content we increase the portability of our approach to other collections where “social” features might not available, e.g., Web search.

The paper 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. We overview related work in Section 5 and conclude the paper in Section 6.

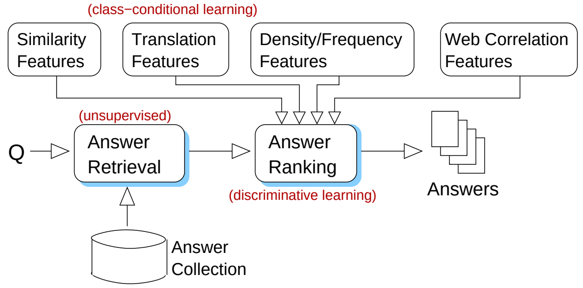

Figure 1: System architecture.

2 Approach

The architecture of the QA system analyzed in the paper, summarized in Figure 1, follows that of the most successful TREC systems. The first component, answer retrieval, extracts a set of candidate answers **A** for 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. Since our focus is on exploring the usability of the answer content, we do not perform retrieval by finding similar questions already answered (Jeon et al., 2005), i.e., our answer collection **C** contains only the site's answers without the corresponding questions answered.

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. The scoring function is a linear combination of four different classes of features (detailed in Section 2.2). This function is the focus of the paper. To answer our 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. Again, since our interest is in investigating the utility of the answer textual content, we use only information extracted from the answer text when learning the scoring function. We do not use any meta information (e.g., answerer credibility, click counts, etc.) (Agichtein et al., 2008; Jeon et al., 2006).

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 learning.

2.1 Ranking Model

Learning with user-generated content can involve arbitrarily large amounts of data. For this reason we choose as a ranking algorithm the Perceptron which is both accurate and efficient and can be trained with online protocols. Specifically, we implement the ranking Perceptron proposed by Shen and Joshi (2005), which reduces the ranking problem to a binary classification problem. The general intuition 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 simply the inner product between the pattern and the weight vector:

$$f_\alpha(\mathbf{x}) = \langle \mathbf{x}, \alpha \rangle \tag{1}$$

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 $f_\alpha(\mathbf{x}_i - \mathbf{x}_j) = f_\alpha(\mathbf{x}_i) - f_\alpha(\mathbf{x}_j)$. Given a margin function $g(i, j)$ and a positive rate τ, if $f_\alpha(\mathbf{x}_i - \mathbf{x}_j) \leq g(i, j)\tau$, an update is performed:

$$\alpha^{t+1} = \alpha^t + (\mathbf{x}_i - \mathbf{x}_j)\tau g(i, j) \tag{2}$$

By default we use $g(i, j) = (\frac{1}{i} - \frac{1}{j})$, as a margin function, as suggested in (Shen and Joshi, 2005), and find τ empirically on development data. Given that there are only two possible ranks in our setting, this function only generates two possible values. For regularization purposes, we use as a final model the average of all Perceptron models posited during training (Freund and Schapire, 1999).

2.2 Features

In the scoring model we explore a rich set of features inspired by several state-of-the-art QA systems. We investigate how such features can be adapted and combined for non-factoid answer ranking, and perform a comparative feature analysis using a significant amount of real-world data. 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, e.g., extracting the dependency-based representation of the text or generalizing words to their WordNet supersenses (WNSS) (Ciaramita and Altun, 2006). We detail each of these feature groups next.

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). We chose this similarity formula because, out 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[4] with the default parameters.

To understand the contribution of our syntactic and semantic processors we compute the above similarity features for five different representations of the question and answer content:

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

Dependencies (D) - the text is represented as a bag of binary syntactic dependencies. The relative syntactic processor is detailed in Section 3. Dependencies are fully lexicalized but unlabeled and we currently extract dependency paths of length 1, i.e., direct head-modifier relations (this setup achieved the best performance).

Generalized dependencies (D_g) - same as above, but the words in dependencies are generalized to their WNSS, if detected.

Bigrams (B) - the text is represented as a bag of bigrams (larger n-grams did not help). We added this view for a fair analysis of the above syntactic views.

Generalized bigrams (B_g) - same as above, but the words are generalized to their WNSS.

[4] http://ir.dcs.gla.ac.uk/terrier

In all these representations we skip stop words and normalize all words to their WordNet lemmas.

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):

$$P(Q|A) = \prod_{q \in Q} P(q|A) \tag{3}$$

$$P(q|A) = (1 - \lambda)P_{ml}(q|A) + \lambda P_{ml}(q|\mathbf{C}) \tag{4}$$

$$P_{ml}(q|A) = \sum_{a \in A} (T(q|a)P_{ml}(a|A)) \tag{5}$$

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 using the maximum likelihood estimator. $P_{ml}(q|A)$ 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-based algorithm implemented in the GIZA++ toolkit[5].

Similarly with the previous feature group, we add translation-based features for the five different text representations introduced above. By moving beyond the bag-of-word representation we hope to learn relevant transformations of structures, e.g., 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).

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.

Same sentence match - number of non-stop question terms matched in a single sentence in the answer.

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

These last two features are computed also for the other four text representations previously introduced (B, B$_g$, D, and D$_g$). Counting the number of matched dependencies is essentially a simplified tree kernel for QA (e.g., see (Moschitti et al., 2007)) matching only trees of depth 2. Experiments with full dependency tree kernels based on several variants of the convolution kernels of Collins and Duffy (2001) did not yield improvements. We conjecture that the mistakes of the syntactic parser may be amplified in tree kernels, which consider an exponential number of sub-trees.

Informativeness - we model the amount of information contained in the answer by counting the number of non-stop nouns, verbs, and adjectives in the answer text that do not appear in the question.

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): $CCP(Q, A) = hits(Q + A)/(hits(Q)\ hits(A)^{2/3})$ where $hits$ returns the number of page hits from a search engine. When a query returns zero hits we iteratively relax it by dropping the keyword with the smallest priority. Keyword priorities are assigned using the heuristics of Moldovan et al. (1999).

Query-log correlation - as in (Ciaramita et al., 2008) we also compute the correlation between question-answer pairs and a search-engine query-log corpus of more than 7.5 million queries, which shares

[5] http://www.fjoch.com/GIZA++.html

roughly the same time stamp with the community-generated question-answer corpus. We compute 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.

3 The Corpus

The corpus is extracted from a sample of the U.S. Yahoo! Answers logs. In this paper we focus on the subset of advice or "how to" questions due to their frequency and importance in social communities.[6] 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 the above corpus we remove the questions and answers of obvious low quality. We implement this filter with a simple heuristic by keeping only questions and answers that have at least 4 words each, out of which at least 1 is a noun and at least 1 is a verb. 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.[7].

Arguably, all these filters could be improved. For example, the first step can be replaced by a question classifier (Li and Roth, 2005). 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 et al., 1993). Each word was morphologically simplified using the morphological functions of the WordNet library[8]. Sentences were annotated with WNSS categories, using the tagger of Ciaramita and Altun (2006)[9], which annotates text with a 46-label tagset. 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. Next, we parsed all sentences with the dependency parser of Attardi et al. (2007)[10]. It is important to realize that the output of all mentioned processing steps is noisy and contains plenty of mistakes, since the data has huge variability in terms of quality, style, genres, domains etc., and domain adaptation for the NLP tasks involved is still an open problem (Dredze et al., 2007).

We used 60% of the questions for training, 20% for development, and 20% for test. The candidate answer set for a given question is composed by one positive example, i.e., its corresponding best answer, and as negative examples all the other answers retrieved in the top N by the retrieval component.

4 Experiments

We evaluate our results using two measures: mean Precision at rank=1 (P@1) – i.e., the percentage of questions with the correct answer on the first position – and Mean Reciprocal Rank (MRR) – i.e., the score of a question is $1/k$, where k is the position of the correct answer. We use as baseline the output of our answer retrieval component (Figure 1). This component uses the BM25 criterion, the highest performing IR model in our experiments.

Table 2 lists the results obtained using this baseline and our best model ("Ranking" in the table) on the testing partition. Since we are interested in the performance of the ranking model, we evaluate on the subset of questions where the correct answer is retrieved by answer retrieval in the top N answers (similar to Ko et al. (2007)). In the table we report

[6] Nevertheless, the approach proposed here is independent of the question type. We will explore answer ranking for other non-factoid question types in future work.

[7] The data will be available through the Yahoo! Webscope program (`research-data-requests@yahoo-inc.com`).

[8] `http://wordnet.princeton.edu`

[9] `sourceforge.net/projects/supersensetag`

[10] `http://sourceforge.net/projects/desr`

	MRR				P@1			
	$N = 10$	$N = 15$	$N = 25$	$N = 50$	$N = 10$	$N = 15$	$N = 25$	$N = 50$
recall@N	26.25%	29.04%	32.81%	38.09%	26.25%	29.04%	32.81%	38.09%
Baseline	61.33	56.12	50.31	43.74	45.94	41.48	36.74	31.66
Ranking	**68.72**±0.01	**63.84**±0.01	**57.76**±0.07	**50.72**±0.01	**54.22**±0.01	**49.59**±0.03	**43.98**±0.09	**37.99**±0.01
Improvement	+12.04%	+13.75%	+14.80%	+15.95%	+18.02%	+19.55%	+19.70%	+19.99%

Table 2: Overall results for the test partition.

results for several N values. For completeness, we show the percentage of questions that match this criterion in the "recall@N" row.

Our ranking model was tuned strictly on the development set (i.e., feature selection and parameters of the translation models). During training, the presentation of the training instances is randomized, which generates a randomized ranking algorithm. We exploit this property to estimate the variance in the results produced by each model and report the average result over 10 trials together with an estimate of the standard deviation.

The baseline result shows that, for $N = 15$, BM25 alone can retrieve in first rank 41% of the correct answers, and MRR tells us that the correct answer is often found within the first three answers (this is not so surprising if we remember that in this configuration only questions with the correct answer in the first 15 were kept for the experiment). The 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-word methods.

Next we see that learning a weighted combination of features yields consistently marked improvements: for example, for $N = 15$, the best model yields a 19% relative improvement in P@1 and 14% in MRR. More importantly, the results indicate that the model learned is stable: even though for the model analyzed in Table 2 we used $N = 15$ in training, we measure approximately the same relative improvement as N increases during evaluation.

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

Iter.	Feature Set	MRR	P@1
0	BM25(W)	56.06	41.12%
1	+ translation(B_g)	61.13	46.24%
2	+ overall match(D)	62.50	48.34%
3	+ translation(W)	63.00	49.08%
4	+ query-log avg(χ^2)	63.50	49.63%
5	+ answer span normalized by A size	63.71	49.84%
6	+ query-log max(PMI)	63.87	50.09%
7	+ same word sequence	63.99	50.23%
8	+ translation(B)	64.03	50.30%
9	+ tfidf(W)	64.08	50.42%
10	+ same sentence match(W)	64.10	50.42%
11	+ informativeness: verb count	64.18	50.36%
12	+ tfidf(B)	64.22	50.36%
13	+ same word sequence normalized by Q size	64.33	50.54%
14	+ query-log max(χ^2)	64.46	50.66%
15	+ same sentence match(W) normalized by Q size	64.55	50.78%
16	+ query-log avg(PMI)	64.60	50.88%
17	+ overall match(W)	64.65	50.91%

Table 3: Summary of the model selection process.

Table 3 summarizes the outcome of our automatic greedy feature selection process on the development set. Where applicable, we show within parentheses the text representation for the corresponding feature. The process is initialized with a single feature that replicates the baseline model (BM25 applied to the bag-of-words (W) representation). The algorithm incrementally adds to the feature set the feature that provides the highest MRR improvement in the development partition. The process stops when no features yield any improvement. The table shows that, while the features selected span all the four feature groups introduced, the lion's share is taken by the translation features: approximately 60% of the MRR

	W	B	B$_g$	D	D$_g$	W +	W +	W + B +	W + B + B$_g$
						B	B + B$_g$	B$_g$ + D	D + D$_g$
FG1 (Similarity)	0	**+1.06**	-2.01	+0.84	-1.75	+1.06	+1.06	+1.06	+1.06
FG2 (Translation)	+4.95	+4.73	+5.06	+4.63	+4.66	+5.80	+6.01	**+6.36**	+6.36
FG3 (Frequency)	+2.24	+2.33	+2.39	+2.27	+2.41	+3.56	+3.56	**+3.62**	+3.62

Table 4: Contribution of NLP processors. Scores are MRR improvements on the development set.

improvement is achieved by these features. The frequency/density features are responsible for approximately 23% of the improvement. The rest is due to the query-log correlation features. This indicates that, even though translation models are the most useful, it is worth exploring approaches that combine several strategies for answer ranking.

Note that if some features do not appear in Table 3 it does not necessarily mean that they are useless. In some cases such features are highly correlated with features previously selected, which already exploited their signal. For example, most similarity features (FG1) are correlated. Because BM25(W) is part of the baseline model, the selection process chooses another FG1 feature only much later (iteration 9) when the model is significantly changed. On the other hand, some features do not provide a useful signal 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. 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 the first two features chosen by our model selection process use information from the NLP processors. The first chosen feature is the translation probability computed between the B$_g$ question and answer representations (bigrams with words generalized to their WNSS tags). The second feature selected measures the number of syntactic dependencies from the question that are matched in the answer. These results provide empirical evidence that coarse semantic disambiguation and syntactic parsing have a positive contribution to non-factoid QA, even in broad-coverage noisy settings based on Web data.

The above observation deserves a more detailed analysis. Table 4 shows the performance of our first three feature groups when they are applied to each of the five text representations or incremental combinations of representations. For each model corresponding to a table cell 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 left part of Table 4 shows that, generally, the models using representations that include the output of our NLP processors (B$_g$, D and D$_g$) improve over the baseline (FG1 and W).[11] However, comparable improvements can be obtained with the simpler bigram representation (B). This indicates that, in terms of individual contributions, our NLP processors can be approximated with simpler n-gram models in this task. Hence, is it fair to say that syntactic and semantic analysis is useful for such Web QA tasks? While the above analysis seems to suggest a negative answer, the right-hand side of Table 4 tells a more interesting story. It shows that the NLP analysis provides *complementary* information to the n-gram-based models. The best models for the FG2 and FG3 feature groups are obtained when combining the n-gram representations with the representations that use the output of the NLP processors (W + B + B$_g$ + D). The improvements are relatively small, but remarkable (e.g., see FG2) if we take into account the significant scale of the evaluation. This observation correlates well with the analysis shown in Table 3, which shows that features using semantic (B$_g$) and syntactic (D) representations contribute the most *on top* of the IR model (BM25(W)).

[11] The exception to this rule are the models FG1(B$_g$) and FG1(D$_g$). This is caused by the fact that the BM25 formula is less forgiving with errors of the NLP processors (due to the high idf scores assigned to bigrams and dependencies), and the WNSS tagger is the least robust component in our pipeline.

5 Related Work

Content from community-built question-answer sites can be retrieved by searching for similar questions already answered (Jeon et al., 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 above approaches. In fact, it is likely that an optimal retrieval engine from social media should combine all these 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.

In the QA literature, answer ranking for non-factoid questions has typically been performed by learning question-to-answer transformations, either using translation models (Berger et al., 2000; Soricut and Brill, 2006) or by exploiting the redundancy of the Web (Agichtein et al., 2001). Girju (2003) extracts non-factoid answers by searching for certain semantic structures, e.g., causation relations as answers to causation questions. In this paper we combine several methodologies, including the above, into a single model. This approach allowed us to perform a systematic feature analysis on a large-scale real-world corpus and a comprehensive feature set.

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. However, both approaches use similarity models at their core because they require the matching of the lexical elements in the search structures. 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.

Our answer ranking framework is closest in spirit to the system of Ko et al. (2007) or Higashinaka et al. (2008). However, the former was applied only to factoid QA and both are limited to similarity, redundancy and gazetteer-based features. Our model uses a larger feature set that includes correlation and transformation-based features and five different content representations. Our evaluation is also carried out on a larger scale. Our work is also related to that of Riezler et al. (2007) where SMT-based query expansion methods are used on data from FAQ pages.

6 Conclusions

In this work we described an answer ranking engine for non-factoid questions built using a large community-generated question-answer collection. On one hand, this study shows 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. On the other hand, we expect the outcome of this process to help several applications, such as open-domain QA on the Web and retrieval from social media. For example, on the Web our ranking system could be combined with a passage retrieval system to form a QA system for complex questions. On social media, our system should be combined with a component that searches for similar questions already answered; this output can possibly be filtered further by a content-quality module that explores "social" features such as the authority of users, etc.

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. While the features that model question-to-answer transformations provide most benefits, we show that the combination is crucial for improvement.

Lastly, we show that syntactic dependency parsing and coarse semantic disambiguation yield a small, yet statistically significant performance increase on top of the traditional bag-of-words and n-gram representation. We obtain these results using only off-the-shelf NLP processors that were not adapted in any way for our task.

References

G. Attardi, F. Dell'Orletta, M. Simi, A. Chanev and M. Ciaramita. 2007. Multilingual Dependency Parsing and Domain Adaptation using DeSR. *Proc. of CoNLL Shared Task Session of EMNLP-CoNLL 2007.*

E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. 2008. Finding High-Quality Content in Social Media, with an Application to Community-based Question Answering. *Proc. of WSDM*.

E. Agichtein, S. Lawrence, and L. Gravano. 2001. Learning Search Engine Specific Query Transformations for Question Answering. *Proc. of WWW*.

A. Berger, R. Caruana, D. Cohn, D. Freytag, and V. Mittal. 2000. Bridging the Lexical Chasm: Statistical Approaches to Answer Finding. *Proc. of SIGIR*.

M. Bilotti, P. Ogilvie, J. Callan, and E. Nyberg. 2007. Structured Retrieval for Question Answering. *Proc. of SIGIR*.

E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. 2001. Data-Intensive Question Answering. *Proc. of TREC*.

P. Brown, S. Della Pietra, V. Della Pietra, R. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2).

M. Ciaramita and Y. Altun. 2006. Broad Coverage Sense Disambiguation and Information Extraction with a Supersense Sequence Tagger. *Proc. of EMNLP*.

M. Ciaramita, V. Murdock and V. Plachouras. 2008. Semantic Associations for Contextual Advertising. 2008. *Journal of Electronic Commerce Research - Special Issue on Online Advertising and Sponsored Search*, 9(1), pp.1-15.

M. Collins and N. Duffy. 2001. Convolution Kernels for Natural Language. *Proc. of NIPS 2001*.

H. Cui, R. Sun, K. Li, M. Kan, and T. Chua. 2005. Question Answering Passage Retrieval Using Dependency Relations. *Proc. of SIGIR*.

M. Dredze, J. Blitzer, P. Pratim Talukdar, K. Ganchev, J. Graca, and F. Pereira. 2007. Frustratingly Hard Domain Adaptation for Parsing. In *Proc. of EMNLP-CoNLL 2007 Shared Task.*

A. Echihabi and D. Marcu. 2003. A Noisy-Channel Approach to Question Answering. *Proc. of ACL.*

Y. Freund and R.E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37, pp. 277-296.

R. Girju. 2003. Automatic Detection of Causal Relations for Question Answering. *Proc. of ACL, Workshop on Multilingual Summarization and Question Answering*.

S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu. 2000. Falcon: Boosting Knowledge for Answer Engines. *Proc. of TREC*.

R. Higashinaka and H. Isozaki. 2008. Corpus-based Question Answering for why-Questions. *Proc. of IJCNLP*.

J. Jeon, W. B. Croft, and J. H. Lee. 2005. Finding Similar Questions in Large Question and Answer Archives. *Proc. of CIKM*.

J. Jeon, W. B. Croft, J. H. Lee, and S. Park. 2006. A Framework to Predict the Quality of Answers with Non-Textual Features. *Proc. of SIGIR*.

J. Ko, T. Mitamura, and E. Nyberg. 2007. Language-independent Probabilistic Answer Ranking. for Question Answering. *Proc. of ACL*.

X. Li and D. Roth. 2005. Learning Question Classifiers: The Role of Semantic Information. *Natural Language Engineering*.

B. Magnini, M. Negri, R. Prevete, and H. Tanev. 2002. Comparing Statistical and Content-Based Techniques for Answer Validation on the Web. *Proc. of the VIII Convegno AI*IA*.

M.P. Marcus, B. Santorini and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn TreeBank. *Computational Linguistics*, 19(2), pp. 313-330.

D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Goodrum, R. Girju, and V. Rus. 1999. LASSO - A Tool for Surfing the Answer Net. *Proc. of TREC*.

A. Moschitti, S. Quarteroni, R. Basili and S. Manandhar. 2007. Exploiting Syntactic and Shallow Semantic Kernels for Question/Answer Classification. *Proc. of ACL*.

S. Robertson and S. Walker. 1997. On relevance Weights with Little Relevance Information. *Proc. of SIGIR*.

R. Soricut and E. Brill. 2006. Automatic Question Answering Using the Web: Beyond the Factoid. *Journal of Information Retrieval - Special Issue on Web Information Retrieval*, 9(2).

L. Shen and A. Joshi. 2005. Ranking and Reranking with Perceptron, *Machine Learning. Special Issue on Learning in Speech and Language Technologies*, 60(1-3), pp. 73-96.

S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal and Y. Liu. 2007. Statistical Machine Translation for Query Expansion in Answer Retrieval. In *Proc. of ACL*.