Improving Question Answering Sentence Ranking by Rank Propagation

Hong-Wei Ng
University of Illinois, Urbana-Champaign
hongwei2@illinois.edu

Xiaoxiao Wang*
Bloomberg L.P.
xwang743@bloomberg.net

Xinyu Zhang
University of Illinois, Urbana-Champaign
xzhan140@illinois.edu

Zeqiu Wu
University of Illinois, Urbana-Champaign
zeqiuwu1@illinois.edu

ABSTRACT

Pointwise ranking is one of the most straightforward ways to rank candidate answer sentences in a QA system and can sometimes give comparable results to more sophisticated approaches. However, it is limited by its inability to consider the interdependency among the candidate answer sentences, which can result in rankings that are unsatisfactory. We describe RankProp, an algorithm that is used as a post-processing step for improving pointwise ranking under the assumption that semantically similar sentences should be ranked closer. If the expected answer type is available from an answer type prediction module, this information together with the entity types extracted from the sentences can be incorporated to further improve the results RankProp can therefore be viewed as way to fix a simple pointwise ranking model by correlating the rankings it produces for a set of candidate answer sentences rather than use structured machine learning to learn a complex ranking model from the start, which can be computationally expensive to train and hard to scale.

KEYWORDS

Ranking; Question answering; Information retrieval

ACM Reference Format:
Hong-Wei Ng, Xiaoxiao Wang, Xinyu Zhang, and Zeqiu Wu. 2018. Improving Question Answering Sentence Ranking by Rank Propagation. In *Proceedings of First Workshop on Knowledge Base Construction, Reasoning and Mining (KBCOM'18).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

State-of-the-art open domain QA systems usually consists of a pipeline of modules performing tasks such as question analysis, document retrieval, answer sentence selection and answer extraction (Figure 1). In this paper, we focus on improving the ranking of candidate answer sentences obtained from a answer sentence selection module that uses pointwise ranking, i.e., each candidate answer sentence is assigned a relevance/ranking score that is used to order them.

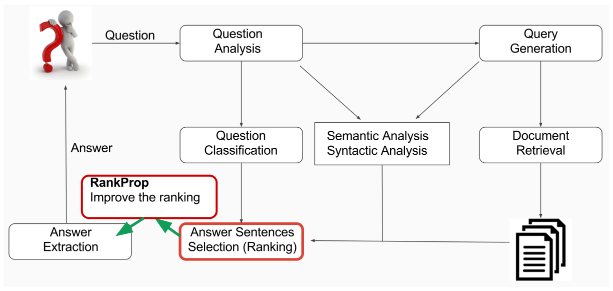

Figure 1: Overview of a typical open domain question answering system. Our proposed algorithm, RankProp, can be inserted seamlessly between a answer sentence selection module that ranks the candidate answers and the answer extraction module.

There are several reasons why one might use a pointwise ranking model: (1) they are straightforward to implement, (2) training data can be obtained readily from clickthrough logs and (3) they can usually be trained on large training sets easily. Furthermore, the result such models produce can sometimes be close or comparable to those by more sophisticated approaches such as pairwise or listwise ranking [7]. However, their main weakness is that they do not consider the interdependency between the objects (in this case candidate answer sentences) being ranked. In particular, noise in the feature representation for the sentences can lead to those containing the correct answer getting assigned low ranking scores even if they are semantically similar to other highly ranked sentences. Conversely, sentences not containing the answer might end up with high ranking scores for the same reasons. The example given in Table 4 and the discussion in section 6.3 elaborates on this point.

Our key observation is that a pointwise ranking model that performs better than random is unlikely to get the majority of its predictions wrong. This means we can exploit self-similarity between the candidate answer sentences to "clean up" the ranking scores and hence improve the ranking. So, rather than use a complex ranking algorithm that can be hard to train robustly with small datasets and computationally expensive to train for large ones, we can use a simple pointwise ranking model to provide the initial

*This work was done when the author was at University of Illinois, Urbana-Champaign.

ranking and then perform structured prediction on the small set of retrieved candidate answer sentences to refine the ranking.

Specifically, we build on the pointwise ranking module of the Jacana QA system by Yao et al. [26, 27], which is a logistic regression classifier trained to predict if a sentence retrieved by the answer sentence selection module contains the answer to a question. We propose an algorithm, RankProp, for refining ranking scores from systems such as those from Jacana QA mentioned above.

RankProp works by propagating the ranking scores of the candidate answer sentences over a similarity graph based on the idea that (1) semantically similar candidate answer sentences should have similar scores, (2) candidate answer sentences containing entities with the type required by the question should be ranked higher and (3) a decent ranking model will get the ranking scores for some candidate answer sentences in the right range and that is sufficient to allow correct ranking scores to propagate to other similar candidate answer sentences. The refined ranking scores are obtained by solving a convex optimization problem. The candidate answer sentences are then re-ranked based on these refined scores.

Our experiments show that using RankProp as a post-processing step consistently improves the ranking obtained from every classifier we tried to generate the pointwise ranking. Note that even though the experiments were run on the feature vectors produced by Jacana QA [26, 27], our algorithm does not actually depend on any specific QA systems. For e.g., RankProp can also be used with the answer sentence selection model from Severyn et al. [20].

Finally, we introduce a question-answer similarity feature calculated from word embeddings [17] and show that adding it to the set of features used by the logistic regression classifier in [26, 27] leads to an improvement in the performance of their ranking module. We also experiment with other classifiers besides logistic regression and demonstrate that this feature also leads to an improvement in the results for them. In summary, our contributions are:

(1) A convex optimization program, RankProp, that incorporates candidate answer sentence similarity and type information to improve existing ranking scores. RankProp can be inserted seamlessly into any QA system that uses a pointwise ranking model to rank candidate answer sentences without modifying the rest of the system.
(2) A novel feature based on question and candidate answer sentence similarity.
(3) Experimental results demonstrating the efficacy of the new feature and our RankProp post-processing algorithm in improving the performance of pointwise ranking models trained using the features described in [26, 27].

2 DEFINITIONS AND PROBLEM

In this work, we focus on the task of improving the ranking quality of a answer sentence selection module in a typical QA system. The input to our proposed algorithm, RankProp, is a question q, expressed in text, a set of candidate answer sentences $\mathcal{A} = \{a_1, \ldots, a_n\}$ for q, and a score for each candidate, $r = (r_1, \ldots, r_n)$ that is used for ranking them.

We assume the scores in r have been normalized to the range $[0, 1]$, where candidate answer sentences with larger scores are more likely to contain the answer to q and hence will be ranked higher. These ranking scores can be generated by any pointwise ranking model. In our experiments we trained a number of classifiers with features computed from question and candidate answer sentence pairs following the approach described in [26, 27]. A classifier outputs the probability that a candidate answer sentence in $\mathcal{A}$ contains the answer to q and that probability is its ranking score.

In addition, we represent each sentence, be it the question or a candidate answer sentence, as a vector encoding its semantic meaning (see section 3.1). These vectors will be used by RankProp to produce a refined version of r, $y = (y_1, \ldots, y_n)$, which can then be used to re-rank the candidate answer sentences in $\mathcal{A}$ to obtain an improved ranking. We also use these feature vectors to construct a new feature that measures the similarity of a question and a candidate answer sentence. Our experiments show that adding this feature to the set of features considered by the ranking module from [26, 27] improves the ranking results (see section 6). Furthermore, if the expected answer type of q, denoted T_q, is provided, for example by a answer type prediction classifier, RankProp can use it along with the entity types found in each of the answers to improve the quality of y (see section 4.3).

3 FEATURES

3.1 Sentence Representation

Word embeddings such as Word2Vec [15] have been shown to be an effective way to represent words. Following that work, Pennington et al. [17] came up with another way to generate word embeddings which they named GloVe. In this work, we use a simple method to represent sentences as vectors. Given a sentence, be it a question q or a candidate answer sentence a_i, we simply take the average of the corresponding 300 dimensional GloVe word vectors for all the non-stopwords in the sentence as its vector representation. These GloVe word vectors are generated from a Wikipedia and Gigaword corpus by Pennington et al. [17] for a 400K vocabulary and therefore is rich enough to cover words in most context. Although this approach appears simple, it has been shown to be effective [10] and gives competitive results compared to more sophisticated methods such as Sentence2Vec [11] that encodes sentences directly. Similarity scores between a question and each of its candidate answer sentences (section 3.2) and between pairs of candidate answer sentences (section 4.2) are then computed using these vectors.

3.2 Question-Answer Similarity

Using the vector representation for sentences described in section 3.1, we compute cosine similarities between a question q and each of its candidate answer sentence, a_i, as a way to encode the similarity between them. The cosine similarity scores will be used in two ways: (1) as a weight to elevate the ranking scores of candidates with types expected by the question (see section 4.3) and (2) as a novel feature that we add to those used by the ranking module of [26, 27] for computing the ranking scores r of the candidate answer sentences retrieved for a question q.

The idea for introducing a measure of question answer similarity is that candidate answer sentences which are similar to the question is likely to contain the answer due to the data redundancy phenomenon. We elaborate on this point in section 4.

4 APPROACH

Our algorithm RankProp post-processes the ranking scores generated by the answer ranking module of a typical QA system based on a few hypotheses:

(1) Candidate answer sentences that are semantically similar to the question and contain entities with the same type as the expected answer type are likely to contain the answer.
(2) Candidate answer sentences that are semantically similar should have similar ranks since they express the same piece of information.
(3) The given ranking score might be noisy because the same piece of information can be expressed in different ways, but in a set of retrieved answers, there is a good chance that some of them have appropriate values.

Hypothesis 1 follows from the work of redundancy-based approach for QA system [6, 13] where it was observed that in a large corpus, data redundancy will result in the same piece of information expressed in different ways. A snippet that contains information close to the question, for e.g., by having many similar words, is therefore likely to contain the expected answer. This is why we compute the cosine similarity between a question and its candidate answer sentences in section 3.2 as a feature for the ranking module, as well as use it as a weighting factor in our algorithm (section 4.3). Furthermore, if a candidate answer sentence contains entities whose type matches the expected answer type, it is more likely that it contains the factoid that answers the question compared to sentences that do not have entities with a matching type.

Hypothesis 2 is intuitive in the sense that if two candidate answer sentence contain the same piece of information, perhaps expressed differently, then they should have similar ranking score as they are equally good/bad candidates for answering the question.

Hypothesis 3 is fairly reasonable as a properly set up ranking module should at least be able to identify a fair number of candidates correctly as it should be performing better than random chance.

RankProp is a convex optimization program. Given ranking scores $r = (r_1, \ldots, r_n) \in [0,1]^n$ for n candidate answer sentences of a question q, their vector representation $v_1, \ldots, v_n$ and optionally the set $\mathcal{M}$ containing the indices of candidates with entities matching the expected answer type T_q, RankProp returns a new set of ranking score $y = (y_1, \ldots, y_n) \in [0,1]^n$ refined based on the hypotheses described above. We describe the various parts of this optimization program and the rationale behind them in the following sections.

4.1 Data Fitting Term

Our goal is to obtain an improved estimate of the ranking scores r. Following hypothesis 3, *some* of the given ranking scores are reliable, therefore ideally we don't want y to deviate too much from r. This intuition gives rise to a penalty term,

$$f_{\texttt{fit}}(y) = \|r - y\|_p$$

where p can be 1 or 2 corresponding to the ℓ_1 or ℓ_2 norm respectively. Our experiments show that in some cases one is better than the other, therefore we left this as a parameter to be tuned.

4.2 Rank Propagation

We use the graph-based semi-supervised learning [31] framework to propagate ranking scores between candidate answer sentences. First, construct a k-nearest neighbor graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, using the vector representation of the candidate answer sentences $v_1, \ldots, v_n$ (see section 3.1) for a question. k is a parameter that we tune and is typically set to 3 or 5 in our experiments. It is data dependent and should be smaller if the answer retrieval module of the QA system tends not to retrieve many candidate answer sentences, but set to larger values otherwise. This is to avoid connecting a candidate to other candidates very different from itself and therefore affect the rank propagation process. Between a pair of candidate answer sentences a_i and a_j that are connected in the graph, we compute their similarity weight w_{ij} using the Gaussian kernel

$$w_{ij} = \exp\left(-\frac{\|v_i - v_j\|_2^2}{2\sigma^2}\right) \tag{1}$$

where σ is the bandwidth of the kernel and also requires tuning. Given these weights, the rank propagation term is defined as

$$\begin{aligned} f_{\texttt{rankprop}}(y) &= \sum_{(i,j)\in\mathcal{E}} w_{ij} \cdot \left(\frac{y_i}{\sqrt{d_i}} - \frac{y_j}{\sqrt{d_j}}\right)^2 \\ &= y^\top D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}} y \\ &= y^\top L y \end{aligned}$$

where

$$W_{ij} = \begin{cases} w_{ij}, & \text{if } (i,j) \in \mathcal{E} \\ 0, & \text{otherwise} \end{cases}$$

$$D_{ij} = \begin{cases} \sum_m W_{im}, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

Here L is the normalized graph Laplacian matrix. As a technical detail, to ensure that L is at least positive semidefinite (PSD), we need to make W symmetric. We ensure this by making the k-nearest neighbor graph undirected. That is, if $(i,j) \in \mathcal{E}$, we also ensure that $(j,i) \in \mathcal{E}$. With this, L is at least PSD, which means our optimization program for propagating ranking scores (see section 4.4) will be a convex optimization problem. The way to interpret this term is that if candidates a_i and a_j are similar, then w_{ij} will be large, therefore their refined ranking scores y_i and y_j will be close in order for them to (roughly) cancel out and minimize this term.

4.3 Using Type Information

If RankProp is given the expected answer type, T_q of the question q, we can obtain the set of candidate answer sentences $\mathcal{M}$ that contains entities with type matching T_q, for e.g., , by running all candidates in $\mathcal{A}$ through a NER tagger and storing the indices of those whose entities' type match T_q. Given $\mathcal{M}$, we want to elevate ranking scores to indicate that they are more likely to contain the correct answer. However, clearly not all sentences containing the required type should have their rank increased as the sentence could just by chance have a matching entity type but the sentence itself could have nothing to do with the question. Therefore, we want to factor in the similarity between the question and the candidate answer sentence when making this adjustment. This results in the following penalty term,

$$f_{\text{type}}(y) = \sum_{i \in \mathcal{M}} w_i^q \cdot |1 - y_i|$$

where w_i^q is defined as the normalized cosine similarity between the question q and the candidate answer sentence's vector v_i. Note that we will later constrain y to be within $[0, 1]$, therefore this term will elevate the ranking scores of candidates towards the maximum possible value, but weighted by their relevancy to the question q.

In our experiments, we only consider three types - Person, Location, and Organization. This term is not used for question and candidate answer sentences whose types do not belong to any of these 3 three types, i.e., $\mathcal{M} = \emptyset$ for those cases.

4.4 RankProp Optimization

Given the initial ranking scores r of a set of candidate answer sentences $\mathcal{A}$ for a question q, pairwise candidate answer sentences similarity scores $\{w_{ij}\}_{(i,j)\in\mathcal{E}}$, and question answer similarity scores $\{w_i^q\}_{i\in\mathcal{M}}$, the optimization problem for RankProp is

$$\begin{array}{ll} \underset{y}{\text{minimize}} & f_{\texttt{fit}}(y) + \alpha f_{\text{rankprop}}(y) + \gamma f_{\text{type}}(y) \\ \text{subject to} & 0 \le y_i \le 1, \; i = 1, \ldots, |\mathcal{A}|. \end{array}$$

$\alpha > 0, \gamma \ge 0$ are weights to be tuned. In the case where T_q is not available, we omit the f_{type} term (i.e., set $\gamma = 0$).

Notice that each of the individual terms in the objective function above is a convex function, so their positive weighted sum is also convex. The constraints are affine inequalities, or more specifically box constraints. Together, they imply RankProp is a convex optimization problem. Importantly, L is a very sparse PSD matrix and the constraints are simple box constraints, so RankProp can be solved efficiently even if $|\mathcal{A}|$ is large. Also, the solution is optimal with respect to the above objective function and constraints.

Our proposed method is summarized as follows:

(1) Represent a question and its candidate answer sentences as vectors following the description in section 3.1.
(2) Use the vector representations of the question and candidate answer sentences to compute the graph laplacian matrix L in section 4.4 and normalized cosine similarity scores between a question and each of its candidate answer sentences.
(3) Obtain initial ranking scores, r, for the candidate answer sentences $\mathcal{A}$, for e.g., by representing question candidate pairs as vectors using the method described in [26] and train a binary classifier (e.g., Logistic Regression) to output the probability that a candidate answer sentence contains the answer to the question as its ranking score.
(4) If the expected answer type T_q of the question q is given, extract the types of entities in the candidate answer sentences using a name entity tagger (e.g., [8]) and let $\mathcal{M}$ be the indices of sentences containing entities with type T_q.
(5) Solve the optimization problem described in section 4.4 using r, and optionally $\mathcal{M}$, to obtain the new ranking scores y.

Although we described methods for performing steps 1 to 4 in their respective sections, the user is free to perform them in any way he/she pleases.

Data	#ques.	#pairs	%pos.	len.	Judgement
TRAIN-ALL	1,229	53,417	12.0	any	automatic
TRAIN	94	4,718	7.4	≤ 40	manual
DEV	82	1,148	19.3	≤ 40	manual
TEST	89	1,517	18.7	≤ 40	manual

Table 1: Summary of the answer sentence selection dataset

5 TREC ANSWER SELECTION DATASET

We use the dataset provided by Wang et al. [25]. This dataset contains labeled sentences from Text Retrieval Conference (TREC) QA track (8-13) data. Specifically, it consists of a set of factoid questions, each having a list of candidate answer sentences that were automatically retrieved from the question's document pool based on non-stop word overlapping counts. These candidates were then compared against the TREC answer pattern to find matched (noisy) positive answers. The correctness of the results for parts of the dataset was judged manually thereafter. In addition, most works only evaluate on questions with both positive and negative answers. They also limit candidate answer sentences to be no longer than 40 words. Details of this dataset are given in Table 1.

We adopt a similar data processing procedure as described in Yao et al. [26], which was originally proposed by Wang et al. [25]. POS tags were labeled using MX-POST [19]; parse tree were built via MST-Parser [14]; named entities recognition was originally performed by Identinfinder [3], but we also ran the named entity recognizer (NER) by Finkel et al. [8] to extract entity types from candidate answer sentences. The expected answer type for each question was manually labeled by referencing the question classification dataset provided by Li and Roth [12].

6 EXPERIMENTS

6.1 Answer Selection Evaluation

We trained four types of classifiers: (1) naive bayes (NB for short), (2) gradient boosting (GB), (3) random forest (RF) and (4) logistic regression (LR) using the original set of 60 features from the Jacana system [26] (labeled **Jacana** in Table 2 and 3) along with an additional similarity feature (labeled **SIM** in Table 2 and 3) calculated from normalized cosine similarity scores between question and candidate answer sentence vectors (see section 3.2) to predict the probability that a candidate answer sentence in $\mathcal{A}$ contains the answer to its corresponding question q. These probabilities are the pointwise ranking scores r described in section 2.

We evaluate the performances of these pointwise ranking models using Mean Average Precision (MAP), which is the mean of the average precision scores for each query, and Mean Reciprocal Rank (MRR). Results without post-processing with RankProp are given in Table 2 and those with RankProp are in Table 3. From Table 2, we see that using the similarity feature along with the features from the Jacana system consistently improves the accuracy of each classifier. Also, we note that the ranking performance of the Jacana system is slightly lower than what was reported in Yao et al. [26] (see Table 2), which could be due to differences in implementation.

[1]Score reported in [26].

		DEV		TEST	
System	Clf	MAP	MRR	MAP	MRR
Rao [18]	CNN	-	-	0.780	0.834
Tan [23]	LSTM	-	-	0.753	0.830
Yao [26][1]	LR	-	-	0.6307	0.7477
Jacana [27][2]	LR	-	-	0.6259	0.7399
Severyn [20]	SVM	-	-	0.678	0.736
Wang [25]	LR	-	-	0.6029	0.6852
Jacana [3]	NB	0.5378	0.6034	0.5506	0.6175
	GB	0.6950	0.7421	0.6288	0.7033
	RF	0.7065	0.7818	0.6200	0.6986
	LR	0.7117	0.7886	0.6237	0.7327
Jacana + SIM	NB	0.5787	0.6464	0.5992	0.6594
	GB	0.7244	0.7787	0.6568	**0.7447**
	RF	**0.7544**	**0.8185**	0.6581	0.7367
	LR	0.733	0.8019	**0.6635**	0.741

Table 2: Results on the QA ranking task by various classifiers without RankProp. SIM is the similarity feature described in section 3.2. See text for details.

		DEV		TEST	
System	Clf	MAP	MRR	MAP	MRR
Jacana+ RankProp	NB	0.5736	0.632	0.5779	0.6445
	GB	0.7164	0.7689	0.6484	0.7176
	RF	0.715	0.8006	0.6261	0.7258
	LR	0.7226	0.7957	0.6413	0.7401
Jacana + SIM + RankProp	NB	0.6012	0.6734	0.6206	0.6826
	GB	0.7389	0.7787	0.6597	0.7667
	RF	**0.7594**	**0.8445**	0.6534	0.7372
	LR	0.74	0.8162	**0.6622**	**0.7723**
$\gamma = 0$, no type info					
Jacana+ RankProp	NB	0.5641	0.6206	0.5779	0.6732
	GB	0.716	0.7649	0.6379	0.7058
	RF	0.715	0.8006	0.6261	0.7258
	LR	0.7101	0.7914	0.6352	0.7548
Jacana + SIM + RankProp	NB	0.5962	0.6576	0.6159	0.7048
	GB	0.7389	0.7787	0.6595	**0.7667**
	RF	**0.7594**	**0.8445**	0.6534	0.7372
	LR	0.739	0.8162	**0.6719**	0.7548

Table 3: Results on the QA ranking task with RankProp.

6.2 Evaluating RankProp

We use RankProp with and without including type information (i.e., $\gamma = 0$) to post-process the ranking scores produced by the classifiers that gave the results in Table 2 to obtain a new ranking for each question and its candidate answer sentences. When applicable, type information from the candidate answer sentences were extracted using the NER tagger by Finkel et al. [8]. Expected answer type when available are obtained manually by referencing the dataset by Li and Roth [12] as mentioned in section 5.

The MAP and MRR for computed from rankings produced by RankProp are given in Table 3. By comparing the results in Tables 2 and 3, we see that the ranking performance of each classifier trained in different settings consistently improved when RankProp is applied as a post-processing algorithm.

In addition, results from Table 3 suggests that the quality of the ranking tends to improve when we include entity type information, although the effect is small. We suspect the reason for the small improvement is because entity type information is not available for most candidate answer sentences as we only consider three entity types (Person, Location, and Organization) in our experiments. That resulted in only around 200 candidate answer sentences in the entire dataset having entities whose type match the expected answer type of their corresponding questions. However, even though the improvement is small, the results suggests that adding type-based information is a promising direction for future work. In particular, we believe that having more data with type information will lead to an improvement in the result.

The best MAP of 0.7723 (Table 3) is obtained using initial ranking from a logistic regression classifier trained on features described in [26] with the question-answer similarity feature we introduced in section 3.2 post-processed using RankProp that considers type information. The best MRR (0.6719) is produced by a similar setup but without including type information (see Table 3). These results suggest that the approach underlying RankProp in general is valid.

[2]Score obtained from running the code from [27] provided by the authors of [26]. Note that it is slightly lower than the reported score in [26].
[3]Scores using the method described in [26] reproduced by us using features from [27] and classifiers from [16].

6.3 Examining a Specific Example

To better understand RankProp, we examine the result of running it on a specific example taken from the TREC QA development set. The initial ranking obtained from a Logistic Regression classifier-based pointwise ranking model similar to the one used to generate the result in Table 2 from Jacana + SIM feature is shown in Table 4. The same ranking refined by RankProp is shown below it.

In the original ranking, we notice that candidate answer sentences ranked 4, 5 and 7 are ranked below sentence 3 even though they contain the answer "Horace Deets'. Presumably this is due to sentence 3 containing the word "CEO" which also appears in the query and the fact that the other sentences are longer and more complex and hence more likely to have "spurious" features extracted by the Jacana QA system [26, 27]. This is a problem of pointwise ranking models alluded to in the introduction.

Next, we examine the refined ranking by RankProp in the same table. We note that the dataset by Li and Roth [12] indicate the expected answer type as PERSON and the NER tagger identified a number of entities (underlined) in the candidate answer sentences as belonging to this type. Therefore, the f_{type} term (see section 4.3) will help to increase the ranking scores of those sentences towards the maximum value of 1. Simultaneously, the f_{prop} term (see section 4.2) will try to ensure the ranking scores of semantically similar sentences are close. Because the sentence originally ranked in position 2 contains entities ("Horace" and "Deets") with the required type and is also fairly similar to the query due to the

Query		
Who is AARP's top official or CEO?		
Score	**Class**	**Original ranking**
0.77945	+	AARP WASHINGTON Horace Deets, president and co-CEO of the AARP, addresses a National Press Club luncheon on the senior advocacy group's legislative priorities and plans for the future .
0.55764	+	"Seniors vote in greater proportion that any other age group , and we vote on issues, not on personalities," said Horace Deets, AARP's executive director.
0.43849	-	Company recently got a new CEO who had said reorganization was likely.
0.33791	+	A <num> -city, <num> -state bus trip featuring AARP President Tess Canja and other volunteers that will start in Philadelphia during the Republican National Convention and end two weeks later in Los Angeles during the Democratic National Convention.
0.29499	+	AARP Executive Director Horace Deets called the bill "good labor, social and economic policy.
0.26472	-	Being the top officer of AARP involves much more than his old job, Perkins acknowledges.
0.22596	+	"If we are over the hill, we certainly have picked up steam going downhill," said Horace Deets, the former Catholic priest-turned-executive director of the <num> million-member AARP .
0.03908	-	The struggle to the top of the hill for the organization is storied, somewhat sordid history.
Score	**Class**	**RankProp post-processed ranking**
1.00000	+	"Seniors vote in greater proportion that any other age group, and we vote on issues, not on personalities," said Horace Deets, AARP's executive director.
0.90247	+	AARP Executive Director Horace Deets called the bill "good labor, social and economic policy.
0.77945	+	AARP WASHINGTON Horace Deets, president and co-CEO of the AARP, addresses a National Press Club luncheon on the senior advocacy group's legislative priorities and plans for the future.
0.57671	+	"If we are over the hill, we certainly have picked up steam going downhill," said Horace Deets, the former Catholic priest-turned-executive director of the <num> million-member AARP.
0.43849	-	Company recently got a new CEO who had said reorganization was likely.
0.34324	-	Being the top officer of AARP involves much more than his old job, Perkins acknowledges.
0.33791	+	A <num> -city, <num> -state bus trip featuring AARP President Tess Canja and other volunteers that will start in Philadelphia during the Republican National Convention and end two weeks later in Los Angeles during the Democratic National Convention.
0.03908	-	The struggle to the top of the hill for the organization is storied , somewhat sordid history.

Table 4: Sample result for a query taken from TREC QA dev set. Original ranking was obtained using the approach described in [27]. A '+' indicate that the sentence contains the answer and '-' if it does not. Words with the expected answer type, "PERSON", detected using a NER tagger are underlined. Ranking scores that changed due to RankProp are bolded. See text for details.

words "AARP" appearing in both of them and "director" being semantically close to "CEO", this sentence's score was elevated to 1. This was also the case for the sentences originally ranked 5th and 7th. The f_{prop} and f_{data} terms however ensured that their refined scores didn't deviate too much from those sentences they are similar to and from their original values, which is why sentence 7's increase was not as large as it is most similar to the sentence originally ranked 4th that retained its score of 0.43849.

Lastly, the score for the sentence containing "Tess Canja" (recgnized as PERSON by the NER) interestingly did not improve. This could be due to its small pairwise similarities (w_{ij}) with other sentences, which is in the order of 10^{-6} to 10^{-10} (compared to 10^{-4} to 10^{-6} for the other sentences), and its fairly small question answer similarity weight, w^q, of 0.59037 (compared to 0.68864 for the top ranked sentence) preventing f_{prop} and f_{type} terms from pulling it away from its original value. This particular example illustrates that RankProp have the desired properties mentioned in section 4.

7 RELATED WORK

Different QA systems use different answer ranking approaches to select the most probable answers from the candidate answer lists. Most prior work on answer selection make use of feature engineering, linguistic tools, or external resources. For e.g., Yih et al. [29] constructed semantic features based on WordNet. Wang and Manning [24] transformed answer selection problem into a syntactical matching between the question/answer parse tree. Yao et al. [26] proposed a Tree Edit Distance (TED) model for aligning an answer sentence tree with the question tree to combine the features extracted from the tree with POS tagging, dependency parsing and named entity recognition features. Severyn et al. [20] designed an automated discriminative tree-edit feature extraction and engineering over parsing trees system. Sultan et al. [21] proposed a joint model for answer sentence ranking and answer extraction that uses both sentence and phrase-level evidence to solve each task.

There were also prior efforts that use deep neural networks to select answers in QA. Yih et al. [30] built a semantic similarity model using convolutional neural networks. Bordes et al. [5] jointly embedded words and knowledge base objects into the same vector space to measure the relevance of question and answer sentences. Iyyer et al. [9] worked on an application of recursive neural networks for factoid QA over paragraphs. Bahdanau et al. [2] proposed a model for text generation, which can be used for answer selection and generation. Agarwal et al. [1] presented a multistage approach to learning to rank candidate answer sentences in a QA system. Bilotti et al. [4] proposed an approach to passage retrieval for QA systems, based on rank-learning techniques, that integrates linguistic and semantic constraint-checking into the retrieval process. Tan et al. [23] proposed an Attentive LSTM model that helps with modeling long sentences.

Unlike the above work, `RankProp` is not a answer selection module but a post-processing algorithm. The goal is to show that "simple" pointwise ranking models based on binary classifiers can be improved by using the similarities of the answer sentences being ranked and the type information extracted from the query and those sentences. While the performance given here is poorer compared to those based on deep neural networks (e.g., [18, 23]), results could potentially improve with better feature representations for the sentences (e.g., Sentence2Vec [11]) and the availablility of more type information from answer type prediction systems such as Yavuz et al. [28] or Sun et al. [22].

8 CONCLUSION

This paper focused on improving the ranking accuracy of the answer sentence selection module in an open-domain QA system. We introduced a similarity feature based on question and answer sentence similarity and proposed an algorithm `RankProp` based on convex optimization that uses the similarities between candidate answer sentences to refine their ranking based on the assumption that sentences that are semantically similar should be ranked closer to each other. Furthermore, if the expected answer type is available, `RankProp` can also take that into account to improve ranking even further. Our experimental results demonstrate the effectiveness of both the new similarity feature and our `RankProp` algorithm.

REFERENCES

[1] Arvind Agarwal, Hema Raghavan, Karthik Subbian, Prem Melville, Richard D. Lawrence, David C. Gondek, and James Fan. 2012. Learning to Rank for Robust Question Answering. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM '12)*. ACM, New York, NY, USA, 833–842. https://doi.org/10.1145/2396761.2396867

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2014). http://arxiv.org/abs/1409.0473

[3] Daniel M Bikel, Richard Schwartz, and Ralph M Weischedel. 1999. An algorithm that learns what's in a name. *Machine learning* 34, 1 (1999), 211–231.

[4] Matthew W. Bilotti, Jonathan Elsas, Jaime Carbonell, and Eric Nyberg. 2010. Rank Learning for Factoid Question Answering with Linguistic and Semantic Constraints. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10)*. ACM, New York, NY, USA, 459–468. https://doi.org/10.1145/1871437.1871498

[5] Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question Answering with Subgraph Embeddings. In *EMNLP*.

[6] Eric Brill, Susan Dumais, and Michele Banko. 2002. An analysis of the AskMSR question-answering system. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 257–264.

[7] Olivier Chapelle and Yi Chang. 2011. Yahoo! learning to rank challenge overview. In *Proceedings of the Learning to Rank Challenge*. 1–24.

[8] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 363–370.

[9] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A Neural Network for Factoid Question Answering over Paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 633–644. http://www.aclweb.org/anthology/D14-1070

[10] Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368* (2016).

[11] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 1188–1196.

[12] Xin Li and Dan Roth. 2006. Learning Question Classifiers: The Role of Semantic Information. *Nat. Lang. Eng.* 12, 3 (Sept. 2006), 229–249. https://doi.org/10.1017/S1351324905003955

[13] Jimmy Lin. 2007. An exploration of the principles underlying redundancy-based factoid question answering. *ACM Transactions on Information Systems (TOIS)* 25, 2 (2007), 6.

[14] Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 91–98.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[17] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation.. In *EMNLP*, Vol. 14. 1532–1543.

[18] Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-Contrastive Estimation for Answer Selection with Deep Neural Networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. ACM, New York, NY, USA, 1913–1916. https://doi.org/10.1145/2983323.2983872

[19] Adwait Ratnaparkhi et al. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*, Vol. 1. Philadelphia, USA, 133–142.

[20] Aliaksei Severyn and Alessandro Moschitti. 2013. *Automatic feature engineering for answer selection and extraction*. Association for Computational Linguistics (ACL), 458–467.

[21] Md Arafat Sultan, Vittorio Castelli, and Radu Florian. 2016. A Joint Model for Answer Sentence Ranking and Answer Extraction. *Transactions of the Association for Computational Linguistics* 4 (2016), 113–125. https://transacl.org/ojs/index.php/tacl/article/view/738

[22] Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. 2015. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1045–1055.

[23] Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 464–473.

[24] Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic Tree-edit Models with Structured Latent Variables for Textual Entailment and Question Answering. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1164–1172. http://dl.acm.org/citation.cfm?id=1873781.1873912

[25] Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA.. In *EMNLP-CoNLL*, Vol. 7. 22–32.

[26] Xuchen Yao, Benjamin Van Durme, Chris Callison-burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *In North American Chapter of the Association for Computational Linguistics (NAACL)*.

[27] Xuchen Yao, Benjamin Van Durme, Chris Callison-burch, and Peter Clark. 2013. Jacana. https://github.com/xuchen/jacana. (2013).

[28] Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. 2016. Improving semantic parsing via answer type inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 149–159.

[29] Scott Wen-tau Yih, Ming-Wei Chang, Chris Meek, and Andrzej Pastusiak. 2013. Question Answering Using Enhanced Lexical Semantic Models.

[30] Scott Wen-tau Yih, Xiaodong He, and Chris Meek. 2014. Semantic Parsing for Single-Relation Question Answering.

[31] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*. 912–919.