Learning to Rank Answers for Definitional Question Answering

Shiyu Wu[1], Xipeng Qiu[1(✉)], Xuanjing Huang[1], and Junkuo Cao[2]

[1] Shanghai Key Laboratory of Intelligent Information Processing School of Computer Science, Fudan University, Shanghai, China
{sywu13,xpqiu,xjhuang}@fudan.edu.cn
[2] Department of Computer Science, Hainan Normal University, Haikou, China
jkcao@qq.com

Abstract. In definitional question answering (QA), it is essential to rank the candidate answers. In this paper, we propose an online learning algorithm, which dynamically construct the supervisor to reduce the adverse effects of the large number of bad answers and noisy data. We compare our method with two state-of-the-art definitional QA systems and two ranking algorithms, and the experimental results show our method outperforms the others.

1 Introduction

Definitional question answering (QA), as an important form of complex QA, has attracted more and more attention recently. Definitional QA looks for extended answers that are composed of pieces of relevant information spread over many documents in a corpus, such as a biography of a person (e.g. "Who is George Bush?"), and the definition of a generic term (e.g. "What is naproxen?") [9].

The development of definitional QA has been boosted by the Text Retrieval Conference (TREC). For a definitional question such as "Who is X" or "What is X", we call "X" **target**. Most definitional QA systems have the following pipeline structure:

1. Use target as query to retrieve the related sentences;
2. Rank the returned candidate sentences;
3. Remove redundant sentences and return top k sentences as answers.

In definitional QA, most works focus on the second step, such as pattern based methods [3,5,10] and centroid vector based methods [1,7]. Xu et al. [10] ranked candidate sentences by RankSVM [6]. Han et al. [4] ranked the candidate sentences from the two points of view: topic and definition.

However, most of these rank methods have two weaknesses: firstly, it is difficult to sample bad answers because the number of bad answers is usually far larger than good answers; secondly, it is hard to judge whether an answer is good or not in an objective way, so the training data is often noisy.

In this paper, we propose an online learning algorithm, which dynamically construct the supervisor on each iteration and assure the quality of the top k

M. Sun et al. (Eds.): CCL and NLP-NABD 2015, LNAI 9427, pp. 326–332, 2015.
DOI: 10.1007/978-3-319-25816-4_26

returned answers, instead of optimizing rank of the whole candidate list. Our learning algorithm is based on Passive-Aggressive algorithm [2], which passively accepts a solution whose loss is zero, while it aggressively forces the new prototype vector to stay as close as possible to the one previously learned.

The rest of the paper is organized as following. We present our algorithm to rank the candidate sentences in Sect. 2 and describe the features in Sect. 3. Then we give our experiments in Sect. 4. Section 5 concludes the paper.

2 Rank Answers with Variant Passive-Aggressive Algorithm

In this section, we propose an online learning algorithm to rank the answers. Given a target x and the set of its associated candidate answers C, we find a subset of $\hat{Y} \subset C$ with size k as the returned answers by

$$\hat{Y} = \arg\max_{Y \subset C} \mathbf{w}^T \Phi(x, Y), \tag{1}$$

where $\Phi(x, Y)$ is a feature vector.

We define the distance between two sets A and B by inverse Jaccard Similarity,

$$\Delta(A, B) = |A \cup B| / |A \cap B|. \tag{2}$$

Assuming that the subset $Y^* \subset C$ concludes all good answers. We wish to learn $\mathbf{w}$ so that $\Delta(\hat{Y}, Y^*)$ is as small as possible.

We use Passive Aggressive (PA) algorithm [2] to find the new weight vector w_{t+1} to be the solution to the following constrained optimization problem in round t.

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \frac{1}{2} ||\mathbf{w} - \mathbf{w}_t||^2 + \mathcal{C}\xi \tag{3}$$

$$\textbf{s.t. } \ell(\mathbf{w}; x_t) <= \xi \textbf{ and } \xi >= 0. \tag{4}$$

where $\ell(\mathbf{w}; x_t)$ is the hinge-loss function, ξ is a slack variable, and $\mathcal{C}$ is a positive parameter which controls the influence of the slack term on the objective function.

Different from standard PA algorithm, we define the loss as,

$$\ell(\mathbf{w}; x) = \begin{cases} 0, & \gamma(\mathbf{w}; x) > \Delta(Y^*, \hat{Y}) \\ \Delta(Y^*, \hat{Y}) - \gamma(\mathbf{w}; x), & \text{otherwise} \end{cases} \tag{5}$$

where

$$\gamma(\mathbf{w}; x) = \mathbf{w}^T \Phi(x, Y^*) - \mathbf{w}^T \Phi(x, \hat{Y}), \tag{6}$$

We abbreviate $\ell(\mathbf{w}; x)$ to ℓ. If $\ell = 0$ then $\mathbf{w}_t$ itself satisfies the constraint in Eq. (3) and is clearly the optimal solution. We therefore concentrate on the case where $\ell > 0$.

Since it is hard to judge whether an answer is good or not in an objective way, we do not use directly the manual answer set Y^* in our learning process. We dynamically construct the supervisor on each iteration.

We define $\theta = \min\{|\hat{Y} - Y^*|, |(C - \hat{Y}) \cap Y^*|\}$ is the minimal number of bad answers in top-k and good answers out of top-k. In each iteration, we build Y^{**} by inserting θ good answers, denoted as P, from out of top-k into top-k, and excluding the same number of bad answers, denoted as Q.

$$Y^{**} = (\hat{Y} - Q) \cup P. \tag{7}$$

Now, we will show how we decide P and Q.

Assuming in the round i, the rank of the s^t is $r_i(s^t)$. After the update of w, the rank of the s^t is $r_{i+1}(s^t)$. We defined the distance of two rankings, $d(r_i, r_{i+1})$ as the sum of un-concordant pairs.

Theorem 1. *Given two rankings r_1, r_2, over $s_1, s_2, \ldots, s_m$, w.l.g. we let $r_1(s_i) = i$, if $r_2(s_i) > r_2(s_j), i < j$, then $d(r_1, r_2) \geq j - i$*

Proof. Consider the rank $r_2^0 : s_1 \ldots s_{i-1}, s_j , s_i, s_{i+1} \ldots s_{j-1} , s_{j+1} \ldots s_m$. Obliviously, $d(r_1, r_2^0) = j - i$ and the set of the un-concordant pairs of r_2^0 is $NC^0 : \{(s_j, s_i), (s_j, s_{i+1}), \ldots, (s_j, s_{j-1})\}$. Assuming that there is a rank $r_2^1 : r_2^1(s_i) > r_2^1(s_j)$, set of the un-concordant pairs of which is denoted as NC^1 and $d(r_1, r_2^1) < j - i$. Then there must be some $K \subseteq [i + 1, j - 1] : \forall k \in K, (s_j, s_k) \in NC^0 \vee (s_j, s_k) \notin NC^1$. However, $\forall k \in K, (s_i, s_k) \in NC^1$, so $|NC^1| \leq |NC^0|$, which is contrast with the assumption.

Theorem 1 can easily be extended to the following case.

Theorem 2. *Given two rankings r_1, r_2, over $s_1, s_2, \ldots, s_m$, w.l.g. we let $r_1(s_i) = i$, if $I, J \subseteq [1, m], \forall i \in I, \forall \in J, i < j$ and $r_2(s_i) > r_2(s_j)$, then $d(r_1, r_2) \geq \sum_{j \in J} j - \sum_{i \in I} i$*

Let the positions of elements of P in $r_i(s^t)$ are $r_i(P_1) < r_i(P_2) < \cdots < r_i(P_\theta)$, and positions of elements of Q are $r_i(Q_1) < r_i(Q_2) < \cdots < r_i(Q_\theta)$, then according to our constraints and the Theorem 2

$$d(r_i(s^t), r_{i+1}(s^t)) \geq \sum_{j=1}^{\theta} (r_i(P_j) - r_i(Q_j)) \tag{8}$$

The lower bound of the distance of two rankings is $\sum_{j=1}^{\theta} (r_i(P_j) - r_i(Q_j))$. We assume $\|w_{i+1} - w_i\|$ will be increased with the increase of the lower bound. The intuition interpretation is if it is asked to change the positions of two adjacent sentences in $r_i(s^t)$, the change of w will be small, and instead, if it is asked to change the positions of two sentences which are the begin and the end of the ranked list, w will change more largely. In order to minimize $\|w_{i+1} - w_i\|$, $\sum_{j=1}^{\theta} (r_i(P_j) - r_i(Q_j))$ should be minimized by minimizing each $r_i(P_j)$ and maximizing each $r_i(Q_j)$. So we get:

P : the top-θ good answers in $C - Y^*$;
Q : the bottom-θ non-answers in $\hat{Y}$.

Similar to [2], we get the update step,

$$\alpha_t = \min\left(\mathcal{C}, \frac{\Delta(Y^{**}, \hat{Y}) - \mathbf{w}_t^T\left(\Phi(\mathbf{x}, Y^{**}) - \Phi(\mathbf{x}, \hat{Y})\right)}{||\Phi(\mathbf{x}, Y^{**}) - \Phi(\mathbf{x}, \hat{Y})||^2}\right). \tag{9}$$

Our final algorithm is shown in **Algorithm** (1).

input : training data set: $(x_n, C_n, Y_n^*), n = 1, \cdots, N$, and parameters: $\mathcal{C}, k$
output: $\mathbf{w}$

Initialize: $\mathbf{w} \leftarrow 0$,;
for $t = 0 \cdots T - 1$ **do**
  pick a sample (x_t, C_t, Y_t^*) from data set;
  calculate $\hat{Y}_t$ and Y^{**}; calculate $\gamma(\mathbf{w}; x_t)$,and$\Delta(Y_t^{**}, \hat{Y}_t)$;
  if $\gamma(\mathbf{w}; x) \leq \Delta(Y_t^{**}, \hat{Y}_t)$ **then**
    calculate α_t by Eq. (9);
    update $\mathbf{w} \leftarrow \mathbf{w} + \alpha_t(\Phi(x_t, Y_t^{**}) - \Phi(x_t, \hat{Y}_t))$;
  end
end

Algorithm 1: Passive-Aggressive Algorithm for Answering Raning

3 Features

For simplicity, we assume that each sentence is independent with others and remove the redundance later, the feature vector can be decomposed by $\Phi(x, Y) = \sum_{s \in Y} \phi(x, s)$.

3.1 Features Based on Language Models

For a candidate sentence s, we can calculate log $P(s|\text{Corpus})$ using language models trained on different corpora. Here we use four corpora: AQUAINT, modified AQUAINT (AQUAINT*), target corpus (TC) and definition corpus (DC).

Aquaint Aquaint consists of newswire text data in English, drawn from three sources and contains roughly 375 million words correlating to about 3GB of data. $P(s|Aquaint)$ is used to estimate the complexity of the sentence.

Aquaint* Aquaint* is the modified version of Aquaint, which replace the named entities and number word with their entity types(PERSON, LOCATION, ORGANIZATION, BASICNAME) and POS tag (CD). $P(s|Aquaint*)$ is used to measure the complexity of sentence after eliminating the effect of different number words and named entity of same type.

Target Corpus(TC) To each target, we build a TC correspondingly. We get top ranked 100 snippets by Google with the target as the query. Parameters are smoothed on Aquaint by Dirichlet smoothing [11].

Definition Corpus(DC) We build a corpus composed of definitional sentence by collection Wikipedia article on the train targets. Because some words like named entity phrase and number word may be high related with the specific target, we rewrite them in the same way as Aquaint*.

We use unigram model to get the four features of the sentence s, $\log P(s|TC)$, $\log P(s|DC)$, $\log P(s|Aquaint)$, $\log P(s|Aquaint*)$, where the models train by TC and DC are Dirichlet smoothed with the general corpus of Aquaint and Aquaint*, respectively.

3.2 Features Based on Dependency Relation Patterns

The syntax of a sentence is also important. For example, appositive structure often appears in the definition sentences. We use Minipar [8] to get a set of dependency relations patterns. First, to each sentence s as training sample, we get a set of triples: ⟨word, relation, word⟩. Then, we use two wildcard IN_TARGET, OUT_TARGET to indicate whether the content word is the target. Thus, we can get 20 most frequent patterns in the form of <IN_TARGET, relation, OUT_TARGET>.

Redundancy Features. Redundancy features are in the form of $\psi(x^t, s^t_{i_j}, s^t_{i_1 \ldots i_{j-1}})$ to the sentence $s^t_{i_j}$ for the target x^t. For each content word tw in target, we test a range $[a, b]$ centered by tw, denoted as r_{tw}, in $s^t_{i_j}$ for it. The word w is in the range r_{tw} if and only if there are at least $a-1$ and at most $b-1$ words between w and tw. We calculate how many content words have been appeared in the previous sentences as following.

$$\frac{|\{w : \exists tw, w \in r_{tw} \vee \exists s^t_{i_{j'}}, 1 \leq j' \leq j-1, w \in s^t_{i_{j'}}\}|}{|\{w : \exists s^t_{i_{j'}}, 1 \leq j' \leq j-1, w \in s^t_{i_{j'}}\}|}$$

We used 3 different range $[1, 5], [6, 10], [10, +\infty]$ to catch features (Tables 2 and 3).

Table 1. Results on TREC 2005 ($k = 12$)

System	F-3 Score
Soft-Pattern (SP)	0.29
Human Interest Model (HIM)	0.30
RankPA	**0.35**

Table 2. F-3 score on the TREC 2006

k	RankPA	RankSVM	Han
10	**0.25**	0.18	0.23
15	**0.29**	0.23	0.24
20	**0.28**	0.24	0.26
25	**0.28**	0.244	0.26
30	**0.26**	0.234	0.26
35	**0.23**	0.194	0.23

Table 3. Recall on the TREC 2006

k	RankPA	RankSVM	Han
10	**0.31**	0.23	0.29
15	**0.39**	0.32	0.33
20	**0.45**	0.37	0.38
25	**0.45**	0.40	0.40
30	**0.46**	0.40	0.42
35	**0.47**	0.42	0.44

4 Experiments

We use three datasets from TREC 2004 $\sim$ 2006, which include 65, 75, 75 definitional questions respectively.

We firstly use Lucene[1] to get at most 200 sentences from Aquaint related to the target, and the query is just the target.

We adopt F-3 score, which used in the TREC definitional question answering task [9].

$$NR = \frac{r}{R} \tag{10}$$

$$NP = \begin{cases} 1, & l < 100 \times (r + a) \\ 1 - \frac{l - 100 \times (r+a)}{1}, & \text{otherwise} \end{cases} \tag{11}$$

$$F\text{-}3 = \frac{10 \times NR \times NP}{9 \times NP + NR} \tag{12}$$

where r is number of vital nuggets in the system response, R is number of vital nuggets in the gold standard, a is number of okay nuggets in the system response and l is length of the system response. Vital nuggets represent the most important facts about the target and should be included. Okay nuggets contribute to relevant information but are not essential.

4.1 Comparison with Other Systems

Kor and Chua [7] gave the results of Soft Pattern model (SP) and Human Interests Model (HIM), which both used questions in TREC 2004 as training data and questions in TREC 2005 as test data. Table 1 shows our method clearly outperforms SP and HIM.

[1] Apache Lucene, http://lucene.apache.org.

4.2 Comparison with Other Ranking Algorithms

To demonstrate the effectiveness of our ranking method, we also compare it with RankSVM [10] and Han [4]. We use the questions in TREC 2005 as training data and TREC 2006 as test data. The targets include PERSON, ORGANIZATION, THING, EVENT.

5 Conclusion

In this paper, we propose an online learning algorithm, which dynamically construct the supervisor on each iteration and assure the quality of the top k returned answers, instead of optimizing rank of the whole candidate list. In the future, we will seek the applications of our method on the ranking problems in other tasks such as summarization.

Acknowledgments. We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by the National Natural Science Foundation of China (61472088, 61363032), the National High Technology Research and Development Program of China (2015AA015408), Shanghai Science and Technology Development Funds (14ZR1403200).

References

1. Chen, Y., Zhou, M., Wang, S.: Reranking answers for definitional qa using language modeling. In: Proceedings of ACL (2006)
2. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. J. Mach. Learn. Res. **7**, 551–585 (2006)
3. Cui, H., Kan, M.: Generic soft pattern models for definitional question answering. In: Proceedings of ACL (2005)
4. Han, K., Song, Y., Rim, H.: Probabilistic model for definitional question answering. In: Proceedings of SIGIR (2006)
5. Hildebrandt, W., Katz, B., Lin, J.: Answering definition questions using multiple knowledge sources. In: Proceedings of HLT-NAACL (2004)
6. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of SIGKDD (2002)
7. Kor, K., Chua, T.: Interesting nuggets and their impact on definitional question answering. In: Proceedings of SIGIR (2007)
8. Lin, D.: Minipar: a minimalist parser. In: Maryland Linguistics Colloquium (1999)
9. Voorhees, E.: Overview of the trec 2004 question answering track. In: Proceedings of TREC (2004)
10. Xu, J., Cao, Y., Li, H., Zhao, M.: Ranking definitions with supervised learning methods. In: Proceedings of WWW (2005)
11. Zhai, C., Lafferty, J.: A study of smoothing methods for language models applied to information retrieval. ACM Trans. Inf. Syst. **22**, 179–214 (2004)