An End-to-End Deep Framework for Answer Triggering with a Novel Group-Level Objective

Jie Zhao
The Ohio State University
zhao.1359@osu.edu

Yu Su
University of California, Santa Babara
ysu@cs.ucsb.edu

Ziyu Guan
Northwest University, China
ziyuguan@nwu.edu.cn

Huan Sun
The Ohio State University
sun.397@osu.edu

Abstract

Given a question and a set of answer candidates, answer triggering determines whether the candidate set contains any correct answers. If yes, it then outputs a correct one. In contrast to existing pipeline methods which first consider *individual* candidate answers separately and then make a prediction based on a threshold, we propose an end-to-end deep neural network framework, which is trained by a novel *group-level* objective function that directly optimizes the answer triggering performance. Our objective function penalizes three potential types of error and allows training the framework in an end-to-end manner. Experimental results on the WIKIQA benchmark show that our framework outperforms the state of the arts by a 6.6% absolute gain under F_1 measure[1].

1 Introduction

Question Answering (QA) aims at automatically responding to natural language questions with direct answers (Heilman and Smith, 2010; Severyn and Moschitti, 2013; Yao et al., 2013; Berant and Liang, 2014; Yih et al., 2015; Sun et al., 2015; Miller et al., 2016; Sun et al., 2016). Most existing QA systems always output an answer for any question, no matter whether their answer candidate set contains correct answers or not (Feng et al., 2015; Severyn and Moschitti, 2015; Yang et al., 2016; Rao et al., 2016). In practice, however, this can greatly hurt user experience, especially when it is hard for users to judge answer correctness. In this paper, we study the critical yet under-addressed *Answer Triggering* (Yang et al., 2015) problem: Given a question and a set of answer candidates, determine whether the candidate set contains any correct answer, and if so, select a correct answer as system output.

The answer triggering problem can be logically divided into two sub-problems: $\mathbf{P}_1$: Build an *individual-level* model to rank answer candidates so that a correct one (if it exists) gets the highest score. $\mathbf{P}_2$: Make a *group-level* binary prediction on the existence of correct answers within the candidate set. Previous work (Yang et al., 2015; Jurczyk et al., 2016) attack the problem via a pipeline approach: First solve P_1 as a ranking task and then solve P_2 by choosing an optimal threshold upon the previous step's highest ranking score. However, the yielded answer triggering performance is far from satisfactory, with F_1 between 32% and 36%. An alternative pipeline approach is to first solve P_2 and then P_1, i.e., first determine whether there's a correct answer in the candidate set and then rank all candidates to find a correct one. However, as we will show using state-of-the-art Multiple Instance Learning (MIL) algorithms in Section 4, P_2 by itself is currently a very challenging task, partly because of the difficulty of extracting features from a set of candidate answers that are effective for answer triggering. Because both P_1 and P_2 performances are far from perfect, the above pipeline approaches also suffer from error propagation (Finkel et al., 2006; Zeng et al., 2015).

We propose Group-level Answer Triggering (GAT), an end-to-end framework for jointly optimizing P_1 and P_2. Our key contribution in GAT is a novel group-level objective function, which aggregates individual-level information and penalizes three potential error types in answer triggering as a group-level task. By optimizing this objective function, we can directly back-propagate the

[1]Our code is available at https://github.com/jiez-osu/answer-triggering.

final answer triggering errors to the entire framework and learn all the parameters simultaneously. We conduct evaluation using the same dataset and measure as in previous work (Yang et al., 2015; Jurczyk et al., 2016), and our framework improves the F_1 score by **6.6%** (from 36.65% to 43.27%), compared with the state of the art.

2 Framework

Notations. Let i and j respectively be the index of question and answer candidate, $l_{i,j}$ be the binary label of the j-th answer candidate for question q_i, and l_i be the group label of the answer candidate set of q_i (1 if it contains any correct answer; 0 otherwise). $m_{i,j}$ denotes an individual-level matching score, measuring how likely question q_i can be correctly addressed by its j-th answer candidate.

The GAT framework is illustrated in Figure 1, which consists of three components: (1) ***Encoder***. Two separate encoders process questions and answer candidates respectively, mapping them from token sequences into two different vector spaces. (2) ***QA Matching***. For each question and answer candidate pair, we concatenate their encoded vectors, and pass it through a feed forward neural network with a binary softmax output layer. The output is an *individual-level* matching score, i.e., $m_{i,j}$. (3) ***Signed Max Pooling***. Max pooling is applied on all the matching scores in a candidate set. During training when each candidate is positively/negatively labeled on whether they can answer the question or not, we use the labels to divide the scores into two disjoint subsets and perform max pooling separately:

$$m_i^+ = \max_{j:l_{i,j}=1} m_{i,j}, \qquad m_i^- = \max_{j:l_{i,j}=0} m_{i,j},$$

where m_i^+ is the maximum score among correct answers (if there's any) and m_i^- is that among wrong ones. At testing time when labels are unavailable, it reduces to normal max pooling and pools a single score $m_i = \max_j m_{i,j}$. The answer triggering prediction is then made by comparing m_i with a predefined threshold (0.5) to decide whether to return the top-scored answer candidate to the user.

The GAT framework design is generic in that the Encoder component can be instantiated with different network architectures. In this paper, we implement it with Bidirectional RNNs (Bi-RNN) (Schuster and Paliwal, 1997) with GRU cells (Cho et al., 2014), and use the temporal average pooling

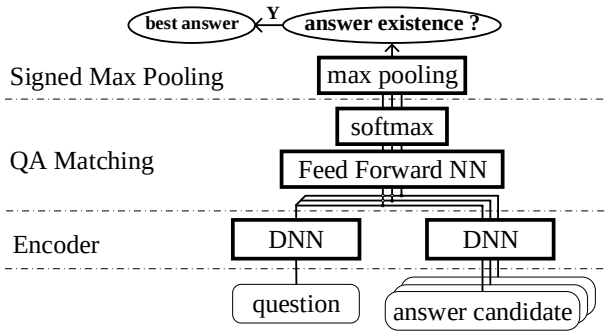

Figure 1: GAT: An end-to-end deep framework to be trained with a novel group-level objective function. Rounded rectangles at the bottom represent input data.

over the hidden states as the encoding representation. We choose Bi-RNN mainly because of its good performance in many QA problems (Wang and Nyberg, 2015; Wang et al., 2016).

2.1 Learning

The cost function for negative groups (answer candidate sets without correct answers) and positive groups (those with correct answers) are treated differently. For each negative group, the highest QA matching score is penalized by a hinge loss:

$$O_1 = \frac{1}{N_{\text{neg}}} \sum_{i:\, l_i=0} \max(0, d^- - (0.5 - m_i^-)),$$

where the maximum matching score m_i^- is compared with 0.5, a fixed threshold for our framework. The variable d^- here, as well as d^+ and $d^\pm$ that will appear shortly after, are all margin hyperparameters. O_1 is normalized by N_{neg}, which is the number of negative groups (with $l_i = 0$). We use O_1 to reduce false-positive answer existence predictions by penalizing the top matching score that is not safely below the 0.5 threshold.

For a positive group, it is more complicated because answer triggering prediction can have the following two error types: (1) the top matching score is below the threshold, or (2) the top ranked answer candidate is a wrong answer. We design loss terms O_2 and O_3 to penalize these two types of error, respectively. O_2 is a hinge loss that penalizes the case where the highest score among the correct answers in a group is not large enough to signify answer existence. O_3 is to penalize the case where the highest score is obtained by an incorrect candidate answer. Formally:

$$O_2 = \frac{1}{N_{\text{pos}}} \sum_{i:\, l_i=1} \max(0, d^+ - (m_i^+ - 0.5))$$

$$O_3 = \frac{1}{N_{\text{pos}}} \sum_{i:\, l_i=1} \max(0, d^\pm - (m_i^+ - m_i^-))$$

Finally, the overall objective function in Equation 1 is a linear combination of the three loss terms and a standard ℓ_2-regularization. Θ denotes all the trainable parameters in the framework. α, β and λ are hyper-parameters.

$$\mathcal{O} = O_1 + \alpha O_2 + \beta O_3 + \lambda \|\Theta\|^2 \quad (1)$$

2.2 A Naive Objective Baseline

For comparison, we provide an alternative objective formulation, which equivalently treats positive and negative groups, and does not explicitly penalize cases where an incorrect candidate answer obtains the highest QA matching score in a positive group.

$$O_2^* = \frac{1}{N_{\text{pos}}} \sum_{i:\, l_i=1} \max(0, d^+ - (m_i - 0.5)) \quad (2)$$
$$O_1^* = O_1; \quad \mathcal{O}^* = O_1^* + \alpha^* O_2^* + \lambda^* \|\Theta\|^2$$

Here d^+ is a margin and α^*, λ^* are weights. We hypothesize this formulation will work worse than the objective in Equation 1, and will use experiments to verify it.

3 Experiments

3.1 Dataset

We use the WIKIQA dataset (Yang et al., 2015) for evaluation. It contains 3,047 questions from Bing query logs, each associated with a group of candidate answer sentences from Wikipedia and manually labeled via crowdsourcing. Several intuitive features are also included in WIKIQA: two word matching features (IDF-weighted and unweighted word-overlapping counts between questions and candidate answers, denoted as `Cnt`), the length of a question (`QLen`), and the length of a candidate answer (`SLen`). As in previous works, we also test the effect of these features, by combining them with other features as input into the Softmax layer in our framework. We use the standard 70% (train), 10% (dev), and 20% (test) split of WIKIQA. We also use the same data pre-processing steps for fair comparison: Truncate questions and sentences to a maximum of 40-token long and initialize the 300-dimensional word vectors using pretrained word2vec embedding (Mikolov et al., 2013).

3.2 Implementation Details

We implement our full framework using TensorFlow (Abadi et al., 2016) and train it using the AdaDelta optimizer (Zeiler, 2012) with learning rate 0.1 and decay factor 0.95. Dropout is used during training to prevent overfitting. The default threshold in Signed Max Pooling is set at 0.5. We select the hyper-parameters using the dev set and set α=1.2, β=1.0, d^+=0.2, d^-=0.3, $d^\pm$=0.5, λ=$1e^{-4}$. The RNN's hidden state size is 200 in both directions. The feed-forward network in *QA Matching* has two layers of 400 hidden units.

3.3 Evaluation Metrics

We use precision, recall, and F_1, defined in the same way as in previous work. A question is treated as a positive case only if it contains one or more correct answers in its candidate set. For the prediction of a question, only the candidate with the highest matching score is considered. A true positive prediction shall meet two criteria: (1) the score is above a threshold (0.5 for our framework; tuned on dev set in other work), and (2) the candidate is labeled as a correct answer to the question.

3.4 Results

a. Comparison with Baselines

We evaluate the effectiveness of the proposed GAT framework by comparing with several baseline models. To the best of our knowledge, there has only been limited work so far on answer triggering, and they are the first two baselines below. (1) Yang et al. (2015) propose CNN-Cnt, which is a combination of the CNN model from Yu et al. (2014) and two `Cnt` features. We use their best reported result which is achieved when CNN-Cnt is combined with `QLen` features. (2) Jurczyk et al. (2016) extend the previous work with various network structures and add some more sophisticated features. Here we compare with their best model on WIKIQA, which is a CNN model combined with carefully designed tree-matching features, extracted from expensive dependency parsing results. (3) We include a third **Naive** baseline where the objective function in Equation 2 is used to train our architecture in Figure 1. Due to space limits, we show its best result obtained among various feature combinations.

The results are summarized in Table 1.

We can see that GAT combined with `Cnt` features improves the F_1 score from Yang et al. (2015) and Jurczyk et al. (2016) by around **11.1%** and **6.6%** (from 32.17 and 36.65 to 43.27), which shows the effectiveness of our framework. We denote this configuration as our *full* framework. Through the comparison between Naive and GAT,

Model	Prec	Rec	F1
(Yang et al., 2015)	27.96	37.86	32.17
(Jurczyk et al., 2016)	29.43	48.56	36.65
Naive +Cnt	27.36	48.84	35.07
GAT	32.70	48.59	39.09
GAT +Cnt (Full)	33.54	60.92	**43.27**
GAT +Cnt+QLen	33.12	59.09	42.45
GAT +Cnt+SLen	28.03	64.60	39.10
GAT +All	31.35	58.82	40.90

Table 1: Results on the test set.

we can see that our proposed objective function has a great advantage over the Naive one which does not model the complexity of answer triggering for positive candidate sets. Different from Yang et al. (2015)'s results, combining with the QLen feature does not further improve the performance in our case, possibly because we choose Bi-RNN as our encoder, which may capture some question characteristics better than a length feature.

b. Framework Breakdown

Now we conduct further analysis in order to better understand the contribution of each component in our full framework. Since the code from (Yang et al., 2015) is available, we use it (rather than (Jurczyk et al., 2016)) to assist our analysis.

We first test a variant of our full framework by replacing the Encoder and QA Matching component with the CNN based model from (Yang et al., 2015)[2], denoted as **GAT** w/ CNN, and train it with our objective. From the first two rows in Table 2, we observe that: (1) Using our current design Bi-RNN and feed-foward NN improves from 35.03% to 43.27%, in comparison with the CNN based model, partly because their CNN only consists of one convolution layer and one average pooling layer. However, we leave more advanced encoder and QA matching design for future work, and anticipate that more complex CNN based models can achieve similar or better results than our current design, as in many other QA-related work (Hu et al., 2014; He and Lin, 2016). (2) Compared with the best result from (Yang et al., 2015) in Table 1, training the CNN based model end-to-end using our objective improves from 32.17% to 35.03%. *This directly shows an end-to-end learning strategy works better than the pipeline approach in (Yang et al., 2015).*

Now we detach the Encoder component **ENC** from our end-to-end full framework. To obtain semantic vectors of questions and candidate answers as input to the subsequent QA Matching component, we leverage Yang et al.(2015)'s released code to train the Encoder component (with CNN) through their well-tuned individual-level optimization, and use their learnt semantic vectors. Then our framework without ENC, i.e., **-ENC**, is trained and tested as before. We further detach the QA matching component **QAM** in a similar way: We directly use the matching score between a question and a candidate answer obtained by Yang et al. (2015), and concatenate it with Cnt features as input to the Softmax layer, which is our framework without ENC or QAM, denoted as **-ENC -QAM**, and trained by our group-level objective. By comparing them with our end-to-end frameworks on both dev and test sets, we can see that it is beneficial to jointly train the entire framework.

[2]Where the QA matching score is obtained first through CNN encoding and then a bilinear model.

	Framework	F_1 score	
		dev	test
End-to-End	Full	44.63	**43.27**
	GAT w/ CNN	39.67	35.03
Pipeline	**-ENC**	39.13	33.42
	-ENC -QAM	38.69	33.20

Table 2: GAT framework breakdown. All variants are trained with our proposed objective function (Equation 1).

3.5 Error Analysis

We now demonstrate some typical mistake types made by our framework to inspire future improvements.

Q: *What city was the convention when Gerald Ford was nominated?*
A: *Held in Kemper arena in Kansas City , Missouri , the convention nominated president Gerald Ford for a full term, but only after narrowly defeating a strong challenge from former California governor Ronald Reagan.*

In this case, **A** is correct, but our framework made a false negative prediction. Although already being the highest ranked in a set of 4 candidate answers, **A** only got a score of 0.134, possibly due to its complicated semantic structure (attribute clause) and the extra irrelevant information (defeating Reagan).

Q: *What can SQL 2005 do?*
A1: *Microsoft SQL server is a relational database management system developed by Microsoft.*

A2: *As a database , it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a [TRUNCATED END]*

The incorrect answer **A1** is ranked higher than the correct answer **A2**, both with scores above 0.5. This is a false positive case, with incorrect ranking as well. Possible reasons are that the detailed functionality of SQL explained in **A2** is hard to be captured and related to the question, and **A2** gets truncated to 40 tokens long in our experiments. On the other hand, the "database management system" phrase in **A1** sounds close to an explanation of functionality, if not carefully distinguished.

Both cases above show that the semantic relation between a question and its answer is hard to capture. For future research, more advanced models can be incorporated in the *Encoder* and *QA Matching* components of our framework.

4 Related Work

Answer Selection. Answer selection (a.k.a., answer sentence selection) is the task of assigning answer candidates with individual-level ranking scores given a question, which is similar to P1 defined in Section 1. Existing QA systems based on answer selection just select the top-scored candidate as answer, without considering the possibility that the true answer doesn't even exist. However, many neural network models recently explored in the answer existence literature (Hu et al., 2014; Wang and Nyberg, 2015; Feng et al., 2015) could be utilized for answer selection as well in the future. For example, Tan et al. (2016) explore the respective advantages of different network architectures such as Long Short-Term Memory Networks (LSTMs) and CNNs. They also develop hybrid models for answer selection. Various attention mechanisms have been proposed such as (Wang et al., 2016) for RNNs and (Yin et al., 2015; dos Santos et al., 2016) for CNNs. Answer selection is also formulated as a sentence similarity measurement problem (He and Lin, 2016; He et al., 2015) or a pairwise ranking problem as in (Severyn and Moschitti, 2015; Yang et al., 2016; Rao et al., 2016).

Multiple Instance Learning We have briefly mentioned MIL (Babenko et al., 2011; Amores, 2013; Cheplygina et al., 2015) in Section 1. Many MIL algorithms can not be directly applied for answer triggering, because individual-level annotations and predictions are often assumed unavailable and unnecessary in MIL (Maron and Lozano-Pérez, 1998; Babenko et al., 2011; Amores, 2013; Cheplygina et al., 2015), but not in the answer triggering setting, where the correctness of each answer candidate is annotated during training and needs to be predicted during testing. We experimented with two popular MIL algorithms that explicitly discriminate individual-level labels: MI-SVM(Andrews et al., 2003) and Sb-MIL (Bunescu and Mooney, 2007) implemented in one of the state-of-the-art MIL toolkits (Doran and Ray, 2014), where we represented each question/answer with encoder vectors as in Section 3.4. Unfortunately, both algorithms predict no correct answer exists for any question, possibly because the training data are biased towards negative groups and the input features are not effective enough. This indicates that using MIL for answer triggering is challenging and still open for future research.

5 Conclusion

In conclusion, we address the critical answer triggering challenge with an effective framework based on deep neural networks. We propose a novel objective function to optimize the entire framework end-to-end, where we focus more on the group-level prediction and take into account multiple important factors. In particular, the objective function explicitly penalizes three potential errors in answer triggering: (1) false-positive and (2) false-negative predictions of the existence of a correct answer, as well as (3) ranking incorrect answers higher than correct ones. We experimented with different objective function settings and show that our GAT framework outperforms the previous state of the arts by a remarkable margin.

Acknowledgments

We would like to thank the anonymous reviewers for valuable comments. The computing resources in this work are supported by Ohio Supercomputer Center (Center, 1987) and the National Science Foundation under Grant No. CNS-1513120. The third author is supported by the National Natural Science Foundation of China (Grant Nos. 61522206, 61672409, 61373118), the Major Basic Research Project of Shaanxi Province (Grant No. 2017ZDJC-31) and the Science and Technology Plan Program in Shaanxi Province of China (Grant No. 2017KJXX-80).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Jaume Amores. 2013. Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, 201:81–105.

Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. 2003. Support vector machines for multiple-instance learning. *Advances in neural information processing systems*, pages 577–584.

Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. 2011. Robust object tracking with online multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1619–1632.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *ACL*, pages 1415–1425.

Razvan C Bunescu and Raymond J Mooney. 2007. Multiple instance learning for sparse positive bags. In *Proceedings of the 24th international conference on Machine learning*, pages 105–112. ACM.

Ohio Supercomputer Center. 1987. Ohio supercomputer center. http://osc.edu/ark:/19495/f5s1ph73.

Veronika Cheplygina, David MJ Tax, and Marco Loog. 2015. Multiple instance learning with bag dissimilarities. *Pattern Recognition*, 48(1):264–275.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Gary Doran and Soumya Ray. 2014. A theoretical and empirical analysis of support vector machine methods for multiple-instance classification. *Machine Learning*, 97(1-2):79–102.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 813–820. IEEE.

Jenny Rose Finkel, Christopher D Manning, and Andrew Y Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626. Association for Computational Linguistics.

Hua He, Kevin Gimpel, and Jimmy J Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*, pages 1576–1586.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of NAACL-HLT*, pages 937–948.

Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019. Association for Computational Linguistics.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050.

Tomasz Jurczyk, Michael Zhai, and Jinho D Choi. 2016. Selqa: A new benchmark for selection-based question answering. *arXiv preprint arXiv:1606.08513*.

Oded Maron and Tomás Lozano-Pérez. 1998. A framework for multiple-instance learning. *Advances in neural information processing systems*, pages 570–576.

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*, pages 3111–3119.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

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*, pages 1913–1916. ACM.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *EMNLP*, volume 13, pages 458–467.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM.

Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. 2016. Table cell search for question answering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 771–782. International World Wide Web Conferences Steering Committee.

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*, pages 1045–1055. ACM.

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*.

Bingning Wang, Kang Liu, and Jun Zhao. 2016. Inner attention based recurrent neural networks for answer selection. In *The Annual Meeting of the Association for Computational Linguistics*.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *ACL*, pages 707–712.

Liu Yang, Qingyao Ai, Jiafeng Guo, and W Bruce Croft. 2016. anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 287–296. ACM.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP*, pages 2013–2018. Citeseer.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *HLT-NAACL*, pages 858–867. Citeseer.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*, pages 1753–1762.