Beihang-MSRA at SemEval-2017 Task 3: A Ranking System with Neural Matching Features for Community Question Answering

Wenzheng Feng†, Yu Wu†, Wei Wu‡, Zhoujun Li†*, Ming Zhou‡
†State Key Lab of Software Development Environment, Beihang University, Beijing, China
‡ Microsoft Research, Beijing, China
{wuyu,lizj,wenzhengfeng}@buaa.edu.cn {wuwei,mingzhou}@microsoft.com

Abstract

This paper presents the system in SemEval-2017 Task 3, Community Question Answering (CQA). We develop a ranking system that is capable of capturing semantic relations between text pairs with little word overlap. In addition to traditional NLP features, we introduce several neural network based matching features which enable our system to measure text similarity beyond lexicons. Our system significantly outperforms baseline methods and holds the second place in Subtask A and the fifth place in Subtask B, which demonstrates its efficacy on answer selection and question retrieval.

1 Introduction

In task 3 of SemEval 2017, participants are required to address typical problems in modern CQA forums. We participate two subtasks: question-comment similarity (Subtask A) and question-question similarity (Subtask B). In Subtask A, given a question and 10 comments in its comment thread, one is required to re-rank the 10 comments according to their relevance with the question. Subtask B gives a question and asks participants to re-rank 10 related questions according to their similarity to the input question.

The challenge of both subtasks is that two natural language sentences often express similar meanings with different but semantically related words, which results in semantic gaps between them. To bridge the semantic gaps, we build a ranking system with a variety of features. In addition to traditional NLP features such as tf-idf (Salton and Buckley, 1988), the longest common subsequence (Allison and Dix, 1986), translation models (Jeon et al., 2005), and tree kernels (Schlkopf et al., 2003; Collins and Duffy, 2002; Moschitti, 2006), which match sentences based on word overlap, syntax (tree kenerls), and word-word translations (translation models), we also introduce neural network based matching models into the system as features. The neural matching features, including a long short term memory network (LSTM) (Schuster and Paliwal, 1997) and a 2D matching network which is a variant of our model in (Wu et al., 2016), can extract high level matching signals from distributed representations of the sentences and capture their similarity beyond lexicons. We also design some specific features for each subtask. All the features are combined as a ranking model by a gradient boosted regression tree which is implemented by Xgboost (Chen and Guestrin, 2016). Our system significantly outperforms baseline methods on the two subtasks. On Subtask A, it holds the second place and is comparable with the best system. On Subtask B, it holds the fifth place. The results demonstrate that our system can alleviate the semantic gaps in the tasks of CQA and effectively rank relevant comments and similar questions to high positions.

2 System Description

Our system is built under a learning to rank framework (Liu et al., 2009). It takes a question and a group of candidates (comments or related questions) as input, and outputs a ranking list of the candidates based on scores of question-candidate pairs. The ranking scores are calculated in three steps: text preprocessing, feature extraction, and feature combination. In preprocessing, we replace special characters and punctuations with spaces, normalize all letters to their lowercase, remove stop-words, and conduct stemming and syntax analysis. Subsequently, we extract a variety of fea-

* Corresponding Author

tures from text pairs including traditional NLP features and neural matching features for both subtasks and some task-specific features. Finally, we feed the features to a ranking model which is trained under a pairwise loss using the training data provided in the subtasks to calculate the ranking scores.

In the following, we will describe details of preprocessing, features, and feature combination.

2.1 Preprocessing

We exploit NLTK toolkit (Loper and Bird, 2002) to conduct stemming, tokenization, and POS tagging. We use Stanford PCFG parser (Klein and Manning, 2003) to get the parse tree of each sentence.

2.2 Traditional NLP Features

The following features are designed based on words and syntactic analysis.

Tf-idf cosine: each piece of text is converted to a one hot representation weighted by tf-idf values, where tf is the term frequency in the text, and idf is calculated using the unannotated Qatar corpora (Nakov et al., 2017). The cosine of representations of the two pieces of text is used as a feature.

Longest common subsequence: we measure the lexical similarity of each text pair with the term-level longest common subsequence (LCS) (Allison and Dix, 1986). The length of LCS is normalized by dividing the maximum length of the two pieces of text.

Word overlap: we calculate the normalized count of common ngrams (n=1,2,3) and nouns.

Tree kernels: tree kernels are similarity functions used to measure the syntactic similarity of a text pair. We compute the subtree kernel (ST) (Schlkopf et al., 2003), the subset tree kernel (SST) (Collins and Duffy, 2002), and the partial tree kernel (PTK) (Moschitti, 2006) on the parse trees of a text pair.

Translation probability: we learn word-to-word translation probabilities using GIZA++ [1] with the unannotated Qatar Living data. In training, we regard questions as source language and their answers as target language. Following (Jeon et al., 2005), we use translation probability $p(\text{qusetion A}|\text{question B})$ and $p(\text{comment}|\text{question})$ as features for a question-pair and a question-comment pair respectively.

In Subtask A, we compute the features on both (question body, comment) and (question subject, comment), and in Subtask B, we compute the features on (question body, question body) and (question subject, question subject).

2.3 Neural Matching Features

In addition to the traditional NLP features, we also use neural matching features to measure text similarity based on their distributed representations. These neural network based models have proven their effectiveness in previous works (Zhang et al., 2016; Fang et al., 2016; Wu et al., 2016; Zhao et al., 2016).

Word embedding cosine: we employ a pre-trained word embedding from `https://github.com/tbmihailov/semeval2016-task3-cqa`, where the dimensionality of word vectors is 200. We average the embedding of words in a piece of text as its representation, and compute the cosine of the representations of two pieces of text as a feature.

Bi-LSTM: long short term memory (LSTM) is an advanced type of recurrent neural network which leverages memory cells and gates to learn long-term dependencies within a sequence (Hochreiter and Schmidhuber, 1997). We use a bidirectional LSTM (bi-LSTM) with a multi-layer perceptron (MLP) to calculate a matching score for a text pair as a feature.

Specifically, given a text pair (S_x, S_y), the model looks up an embedding table to convert S_x and S_y to $\mathbf{S}_x = [e_{x,1}, ..., e_{x,i}, ..., e_{x,I}]$ and $\mathbf{S}_y = [e_{y,1}, ..., e_{y,i}, ..., e_{y,J}]$ respectively, where $e_{x,i}, e_{y,i}$ are the embeddings of the i-th words of S_x and S_y respectively. Then $\mathbf{S}_x$ and $\mathbf{S}_y$ are encoded in hidden sequences by a bi-LSTM which consists of a forward LSTM and a backward LSTM. The forward LSTM reads $\mathbf{S}_x$ in its order (i.e., from $w_{x,1}$ to $w_{x,I}$) and transforms it to a forward hidden sequence $\{\overrightarrow{h}_{x,i}\}_{i=1}^{I}$. $\forall i \in \{1, \ldots, I\}$, $\overrightarrow{h}_{x,i}$ is defined by:

$$\begin{aligned}
i_i &= \sigma(W^{(i)}e_{x,i} + U^{(i)}h_{x,i-1} + b^{(i)}) \\
f_i &= \sigma(W^{(f)}e_{x,i} + U^{(f)}h_{x,i-1} + b^{(f)}) \\
o_i &= \sigma(W^{(o)}e_{x,i} + U^{(o)}h_{x,i-1} + b^{(o)}) \\
u_i &= \tanh(W^{(u)}e_{x,i} + U^{(u)}h_{x,i-1} + b^{(u)}) \\
c_i &= i_i \otimes u_i + f_i \otimes c_{(i-1)} \\
h_i &= o_i \otimes \tanh(c_i),
\end{aligned}$$

[1]`http://www.statmt.org/moses/giza/GIZA++.html`

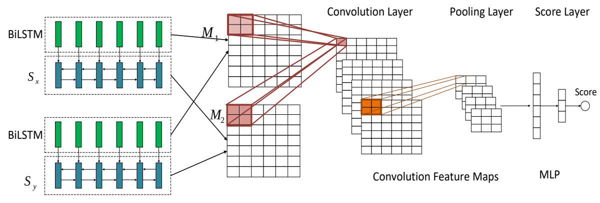

Figure 1: Architecture of 2D matching network

where $\sigma(\cdot)$ is a sigmoid function and $\tanh(\cdot)$ is a hyperbolic tangent function. $W^{(i)}$, $W^{(f)}$, $W^{(o)}$, $W^{(u)}$ $U^{(i)}$, $U^{(f)}$, $U^{(o)}$, $U^{(u)}$, $b^{(i)}$, $b^{(f)}$, $b^{(o)}$, and $b^{(u)}$ are parameters. Similarly, the backward LSTM reads $\mathbf{S}_x$ in its reverse order (i.e., from $w_{x,I}$ to $w_{x,1}$) and transforms it to a backward hidden sequence $\{\overleftarrow{h}_{x,i}\}_{i=1}^{I}$. Then $\forall i \in \{1,\ldots,I\}$, we concatenate $\overrightarrow{h}_{x,i}$ and $\overleftarrow{h}_{x,i}$ as $h_{x,i}$, and then represent S_x as $v_x = average(h_{x,1},...,h_{x,I})$. Following the same procedure, we have v_y as the representation of S_y. Finally, we concatenate (v_x, v_y) as an input of a multi-layer perceptron (MLP) to calculate a score.

2D matching network: the model is a variant of the one proposed in (Wu et al., 2016) which has proven effective on the data of SemEval-2015. The model in (Wu et al., 2016) leverages prior knowledge and performs text matching with multiple channels. In our system, we only use two channels, which means we do not take prior knowledge such as knowledge base (Zheng et al., 2016) and topic information into consideration. The architecture is shown in Figure 1. Given a text pair (S_x, S_y), their word embedding representations $\mathbf{S}_x$, $\mathbf{S}_y$ and their bi-LSTM representations $\{h_{x,i}\}_{i=1}^{I}$ and $\{h_{y,i}\}_{i=1}^{J}$, we compute a word similarity matrix $\mathbf{M}_1 = [m_{1,i,j}]_{I\times J}$ and a sequence similarity matrix $\mathbf{M}_2 = [m_{2,i,j}]_{I\times J}$. $\forall i, j$, the (i,j)-th element of $\mathbf{M}_1$ is defined by

$$m_{1,i,j} = e_{u,i}^{\top} \cdot e_{r,j}. \tag{1}$$

where $e_{u,i}$ is the i-th word embedding of the utterance, and $e_{r,j}$ is the j-th word embedding of the response. The (i,j)-th element of $\mathbf{M}_2$ is defined by

$$m_{2,i,j} = h_{u,i}^{\top}\mathbf{A}h_{r,j}, \tag{2}$$

where $\mathbf{A}$ is a parameter. After that, a convolutional neural network (CNN) takes $\mathbf{M}_1$ and $\mathbf{M}_2$ as input channels, and alternates convolution and max-pooling operations (The system only has one convolution and one pooling layer). Suppose that $z^{(l,f)} = \left[z_{i,j}^{(l,f)}\right]_{I^{(l,f)}\times J^{(l,f)}}$ denotes the output of feature maps of type-f on layer-l, where $z^{(0,f)} = \mathbf{M}_f$, $\forall f = 1, 2$. On convolution layers, we employ a 2D convolution operation with a window size $r_w^{(l,f)} \times r_h^{(l,f)}$, and define $z_{i,j}^{(l,f)}$ as

$$z_{i,j}^{(l,f)} = \sigma\left(\sum_{f'=0}^{F_{l-1}} \sum_{s=0}^{r_w^{(l,f)}} \sum_{t=0}^{r_h^{(l,f)}} \mathbf{w}_{s,t}^{(l,f)} \cdot z_{i+s,j+t}^{(l-1,f')} + b^{l,k}\right), \tag{3}$$

where $\sigma(\cdot)$ is a ReLU (Nair and Hinton, 2010), and $\mathbf{w}^{(l,f)} \in \mathbb{R}^{r_w^{(l,f)} \times r_h^{(l,f)}}$ and $b^{l,k}$ are parameters of the f-th feature map on the l-th layer, and F_{l-1} is the number of feature maps on the $(l-1)$-th layer. A max pooling operation can be formulated as

$$z_{i,j}^{(l,f)} = \max_{p_w^{(l,f)} > s \geq 0} \; \max_{p_h^{(l,f)} > t \geq 0} z_{i+s,j+t}. \tag{4}$$

Feature vectors at the last pooling layer are concatenated to form a similarity vector v, which is fed to an MLP to predict the final similarity score.

We learn the bi-LSTM and the 2D matching network by minimizing cross entropy on training data. Let Θ denote the parameters, then the objective function can be formulated as

$$-\sum_{i=1}^{N} \left[l_i log(f(s_{x,i}, s_{y,i})) + (1-l_i) log(1 - f(s_{x,i}, s_{y,i}))\right], \tag{5}$$

where $l_i \in \{0, 1\}$ is a label, $f(s_{x,i}, s_{y,i})$ is the neural network we want to learn, and N is the number of instances in the training data.

We use two data sets to learn the neural networks, which means we obtain two features from each model. The first one is the training data provided by SemEval-2017 task 3, and the other one is 2 million Yahoo! Answer data we crawled, which is released in (Zhang et al., 2016). In both data, question subjects and question bodies are concatenated together. In SemEval-2017 data, comments in Subtask A are annotated as Good, PotentiallyUseful, and Bad, and we treat "Good" as 1 and the others as 0. In Subtask B, each related question is annotated as PerfectMatch, Relevant, and Irrelevant, and we treat "PerfectMatch" and "Relevant" as 1 and "Irrelevant" as "0". The Yahoo Answer data is only used to learn the neural networks for Subtask A, in which we take a question and its best answer as a positive instance, and randomly sample an answer from other questions as a negative instance. The motivation of leveraging external data is that the training data of SemEval-2017 is small, which may cause overfitting in learning of neural networks.

	Subtask A				Subtask B			
	Train			Test	Train			Test
	2016-train	2016-dev	2016-test	2017-test	2016-train	2016-dev	2016-test	2017-test
Original questions	-	-	-	-	267	50	70	88
Related questions	6154	244	327	293	2670	500	700	880
Comments	37848	2440	3270	2930	-	-	-	

Table 1: Statistics of the datasets

2.4 Task Specific Features

The features described above are used in both Subtask A and Subtask B. In addition to them, we also design some specific features for each subtask.

In Subtask A, we design some features based on heuristic rules which might indicate whether a comment is good or not: (i) whether a comment is written by the author of the question. (ii) the length of a comment. (iii) whether a comment contains URLs or email addresses. (iv) whether a comment contains positive or negative smileys, e.g., ;), :), ;(, :(.

In Subtask B, a related question has a metadata field that shows its relative rank in an external search engine by considering its similarity with the original question. We use the relative rank as a feature for subtask B.

2.5 Feature Combination

Since both Subtask A and Subtask B are ranking problems, we learn gradient boosted regression trees using XgBoost (Chen and Guestrin, 2016) as ranking models to combine all features. The ranking models are learned by minimizing pairwise loss on training instances provided by the subtasks.

3 Experiments

3.1 Data Sets and Evaluation Metrics

We used the data sets provided by SemEval-2017 (Nakov et al., 2017). Table 1 gives the statistics. We employed Mean Average Precision (MAP), Average Recall (AveRec), and Mean Reciprocal Rank (MRR) as evaluation metrics.

3.2 Parameter Tuning and Feature Selection

We tuned parameters according to average MAP on 5-fold cross validation (CV) with grid search algorithm. There are three sensitive parameters of XGBoost that should be tuned in training, namely *gamma*, *subsample*, *colsample_bytree*. The best parameters of two subtasks is shown in Table 2.

	Subtask A	Subtask B
gamma	19	10
subsample	0.5	1
colsample_bytree	0.5	0.2
bst:max_depth	10	10
bst:eta	0.01	0.01
scale_pos_weight	0.7	0.7

Table 2: Parameters of XgBoost

	Bi-LSTM	2D MN
Two LSTMs	not shared	shared
Dim. of embedding	200	200
Dim. of hidden states	200	200
# CNN filters	-	8
CNN filter size	-	(3,3)
nodes of MLP	(200,50,2)	(400,50,2)

Table 3: Parameters of neural networks

We adopted Adagrad (Duchi et al., 2011) which is a stochastic gradient descent method to optimize the neural network models. In order to prevent overfitting, we used early-stopping (Lawrence and Giles, 2000) and dropout (Srivastava et al., 2014) with rate of 0.5. In bi-LSTM and 2D matching network (2D MN), the dimensionality of word embedding is 200. Word embedding was initialized by the result of word2vec (Mikolov et al., 2013) trained on unannotated Qatar data (Nakov et al., 2017) and updated in training. We set the initial learning rate and batch size as 0.001 and 30 respectively. The other parameters of the two models are listed in Table 3.

We conducted feature selection by 5-fold CV to filter out useless features for the two subtasks. Our approach is that we first used all features and obtained an MAP on 5-fold CV, then we removed the features one by one and checked how MAP changes. If MAP increased significantly by removing that feature, we removed the feature. The final result is that we preserved all features for Subtask A, and removed neural matching features for Subtask B. Details of feature contributions will be described in Section 3.5.

Apart from the primary submission, we also

	5-fold cross validation			Test-2017		
Features	MAP	AvgRec	MRR	MAP	AvgRec	MRR
All	70.65	88.54	76.17	88.24	93.87	92.34
- traditional NLP features	69.06	87.94	75.16	87.83	93.60	92.73
- tf-idf cosine	70.28	88.21	76.23	87.88	93.75	92.21
- LCS	69.95	88.01	76.15	88.04	93.90	92.21
- word overlap	69.69	88.10	75.41	88.62	94.14	92.97
- tree kernels	69.50	87.98	95.38	87.97	93.84	92.38
- translation probability	69.77	88.25	75.50	87.81	93.77	92.59
- neural matching features	64.81	82.85	71.91	85.06	91.40	91.52
- word embedding cosine	69.90	88.28	76.24	88.31	93.81	92.40
- bi-LSTM	67.57	86.67	74.54	88.02	93.90	92.54
- 2D MN	69.72	88.01	75.86	88.17	94.04	92.50
- meta-data features	68.09	86.75	74.90	86.54	92.58	91.66

Table 4: Subtask A: results of ablation experiments

	5-fold cross validation			Test-2017		
Features	MAP	AvgRec	MRR	MAP	AvgRec	MRR
All	77.13	91.86	83.89	44.78	79.13	49.89
- tf-idf cosine	72.81	87.85	79.91	44.80	78.60	49.89
- LCS	74.25	88.95	80.90	42.93	78.59	46.94
- word overlap	74.04	88.79	80.67	45.19	80.63	49.65
- tree kernels	76.10	90.98	82.88	45.48	80.63	49.65
- translation probability	75.99	90.35	82.20	44.89	79.57	49.18
- meta-data feature	76.14	91.16	82.98	47.00	80.31	50.83
+ neural matching features*	71.32	86.74	78.21	42.77	77.23	45.98
+ word embedding cosine*	74.76	89.31	81.21	43.59	78.76	46.83
+ bi-LSTM*	71.43	86.88	78.39	42.89	77.89	46.65
+ 2D MN*	70.39	85.49	77.53	43.46	78.60	46.71

Table 5: Subtask B: results of ablation experiments. * means we did not use it in our submitted system for its bad performance on CV.

submitted two contrastive results. The only difference is the parameter setting of XgBoost. In the primary submission, we selected the parameters with which our system achieved the best performance on 5-fold CV, while in the two contrastive submissions, we selected two parameter combinations that correspond to the smallest and the second smallest variance of MAP on 5 runs.

3.3 Baseline

We selected the relative rank provided by the search engine, Google, as a baseline method, and denote it as IR baseline.

3.4 Overall results

We show the primary and contrastive results of Subtask A and Subtask B in Table 6 and Table 7 respectively. There is no significant difference between our primary and contrastive results, indicating that the final result is not sensitive to our parameter selection of Xgboost. On subtask A, the primary and contrastive results significantly outperform the baseline method with a big margin. The primary result, achieving 88.24 on MAP, is ranked second in all submitted systems, demonstrating that neural matching features are effective

Submission	MAP	AvgRec	MRR
primary	88.24	93.87	92.34
contrastive1	88.17	93.82	92.17
contrastive2	88.18	93.91	92.45
KeLP (first)	88.43	93.79	92.82
Baseline (IR)	72.61	79.32	82.37

Table 6: Subtask A: results of our system on test set

Submission	MAP	AvgRec	MRR
primary	44.78	79.13	49.88
contrastive1	43.89	79.48	48.18
contrastive2	44.79	79.13	49.89
simbow (first)	47.22	82.60	50.07
Baseline (IR)	41.85	77.59	46.42

Table 7: Subtask B: results of our system on test set

on the task of answer selection. Our improvement is not big on Subtask B, which is only 3 points on MAP score. This is because we only use shallow features on this task and neural matching features are useless according to our experiments. There are two reasons why neural matching fails on this task: (1) training data provided by SemEval-2017 is too small to train a neural network, and our external data only consists of question-answer pairs which does not support learning neural networks for question-question similarity; (2) a question and its question often share most of words and are only different on a small proportion of function words. Neural matching models, however, are not good at capturing such difference.

3.5 Feature Contribution

We conducted ablation experiments on training data with 5-fold CV and on test data to examine the usefulness of features. The conclusion is that traditional NLP features are effective on both subtasks, while neural matching features can only improve the system performance on Subtask A.

In Table 4, we present the results on Subtask A, including our system with all features and the system with one of the features excluded. We can observe that all features are useful on training data, but the system can achieve a better result on test data if we exclude the word overlap feature. Neural matching features are important on Subtask A, with which we obtain 5 point gain on training data and 3 point gain on test data. Meta-data features are also useful, indicating that they are good complementary to the similarity based features.

In Table 5, we show the results of ablation experiments on Subtask B. Neural matching features caused performance drop on this task, therefore we did not include them in our submitted system. Although all the traditional NLP features are useful on training data, *word overlap*, *tree kernels*, and *meta-data feature* hurt the performance on the test data. It is also worth noting that our system can be further improved on the test data if the meta-data feature, i.e., relative rank of Google, is excluded from our system.

4 Conclusion

We developed a ranking system with neural matching features for Subtask A and Subtask B in SemEval-2017. The system holds the second place in Subtask A and the fifth place in Subtask B, which demonstrates its efficacy on answer selection and similar question retrieval.

5 Acknowledgment

Thanks for the valuable comments given by anonymous reviewers and the discussion with Zhao Yan. This work was supported by the National Natural Science Foundation of China (Grand Nos. 61672081,U1636211, 61370126), Beijing Advanced Innovation Center for Imaging Technology (No.BAICIT-2016001), National High Technology Research and Development Program of China (No.2015AA016004),the Fund of the State Key Laboratory of Software Development Environment (No.SKLSDE-2015ZX-16).

References

L Allison and T I Dix. 1986. A bit-string longest-common-subsequence algorithm. *Information Processing Letters* 23(5):305–310.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 785–794.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In *Meeting on Association for Computational Linguistics*. pages 263–270.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(7):2121–2159.

Hanyin Fang, Fei Wu, Zhou Zhao, Xinyu Duan, Yueting Zhuang, and Martin Ester. 2016. Community-based question answering via heterogeneous social network learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, pages 122–128.

S Hochreiter and J Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *ACM International Conference on Information and Knowledge Management*. pages 84–90.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Meeting on Association for Computational Linguistics*. pages 423–430.

Steve Lawrence and C. Lee Giles. 2000. Overfitting and neural networks: Conjugate gradient and backpropagation 1:114–119 vol.1.

Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3(3):225–331.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. In *ACL 2006, International Conference on Computational Linguistics and Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July*. pages 63–70.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems* 26:3111–3119.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *European Conference on Machine Learning*. pages 318–329.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. pages 807–814.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Vancouver, Canada, SemEval '17.

Gerard Salton and Christopher Buckley. 1988. Termweighting approaches in automatic text retrieval. *Information processing & management* 24(5):513–523.

B Schlkopf, K Tsuda, and J Vert. 2003. Fast kernels for string and tree matching. *Advances in Neural Information Processing Systems* 15(1):296.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Yu Wu, Wei Wu, Zhoujun Li, and Ming Zhou. 2016. Knowledge enhanced hybrid neural network for text matching. *arXiv preprint arXiv:1611.04684* .

Kai Zhang, Wei Wu, Fang Wang, Ming Zhou, and Zhoujun Li. 2016. Learning distributed representations of data in community question answering for question retrieval. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, pages 533–542.

Zhou Zhao, Qifan Yang, Deng Cai, Xiaofei He, and Yueting Zhuang. 2016. Expert finding for community-based question answering via ranking metric network learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, pages 3000–3006.

Hao Zheng, Zhoujun Li, Senzhang Wang, Zhao Yan, and Jianshe Zhou. 2016. Aggregating inter-sentence information to enhance relation extraction. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, pages 3108–3114.