A Short Texts Matching Method Using Shallow Features and Deep Features

Longbiao Kang[1], Baotian Hu[1], Xiangping Wu[1], Qingcai Chen[1,*], and Yan He[2]

[1] Intelligent Computing Research Center,
School of Computer Science and Technology,
Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
{klbgyx7,baotianchina,wxpleduole,qingcai.chen}@gmail.com
[2] Zunyi Medical and Pharmaceutical College, Zunyi, China
xheyan@21cn.com

Abstract. Semantic matching is widely used in many natural language processing tasks. In this paper, we focus on the semantic matching between short texts and design a model to generate deep features, which describe the semantic relevance between short "text object". Furthermore, we design a method to combine shallow features of short texts (i.e., LSI, VSM and some other handcraft features) with deep features of short texts (i.e., word embedding matching of short text). Finally, a ranking model (i.e., RankSVM) is used to make the final judgment. In order to evaluate our method, we implement our method on the task of matching posts and responses. Results of experiments show that our method achieves the state-of-the-art performance by using shallow features and deep features.

Keywords: Short Text, Semantic Matching, Word Embedding, Ranking Model.

1 Introduction

Many natural language processing (NLP) tasks (such as, paraphrase identification [11], information retrieval [8]) can be reduced to semantic matching problems. In this paper, we focus on a short text-matching task called short text conversation, firstly defined by Wang et.al [3]. For a given short text such as a post, this task aims to find a massive suitable response from the candidate set. It's a simplified task of modeling a complete dialogue session such as Turing test. For the convenience of description, we give the following example, the post *P* (post), *R+* (positive response), *R1-* (negative response) and *R2-* (negative response).

- *P: 深圳 今天 天气 怎么样？ How is the weather like today in Shenzhen?*
- *R+: 深圳 现在 正 大雨磅礴 。 It's pouring down in torrents now in Shenzhen.*

* Corresponding author.

C. Zong et al. (Eds.): NLPCC 2014, CCIS 496, pp. 150–159, 2014.

- *R1-: 在 深圳 过的 怎么样 ？ How is everything going in Shenzhen?*
- *R2-:今天 么海 天气 不错哦 。 The weather in Shanghai is very good today.*

The semantic matching is a challenging problem, since it aims to find the semantic relevance between two "text objects". Wang et al. [3] describe a retrieval-based model, which uses about 15 shallow matching features. Most of the features are learned from matching models or generated directly from posts and responses. Although the retrieval-based model performs reasonably well on the post-response matching task, it can only capture the word-by-word matching and latent semantic matching features between posts and responses. For retrieval-based model, it is difficult to recognize R1– as a negative response. However, it is easy to recognize R2– as a negative response, only if we add a feature describing whether the named entities from the post and response are the same. Distributed representation (also called word embedding) of text induced from deep learning is well suited for this task, because it contains rich semantic information of text and can model the relevance between different words or phrases. Related works have demonstrated that the embedding-based method can capture rich semantic relevance between "text objects" and performs well on tasks like paraphrase identification [11] and information retrieval [8]. However, they are not powerful enough at handling the subtlety for specific task. For embeddings of "上海"(Shanghai) and "深圳"(Shenzhen) are very close, it is difficult for embedding-based method to recognize R2- as negative response to P.

In this paper, we study the shallow features and deep features for short text matching and try to combine them to improve the performance. The remainder of this paper is organized as follows. Section 2 reviews the relevant works for our task. Section 3 describes our model for this task. Experimental design, comparison and analysis are presented in Section 4. Finally, we make conclusions in the Section 5.

2 Related Works

Previous works often use rule-based or learning-based models for modeling a complete dialogue [4, 5, 13]. These methods require well-designed rules or particular learning algorithms but relatively less training data. Recent years, by leveraging the massive short text data collected by social media and information retrieval techniques, researchers attack this problem from a new angle [2, 10]. Wang, et al. [3] released a short text dataset collected from Sina Weibo and proposed a retrieval-based response model for short-text conversation. This model considers semantic matching between posts and responses, then retrieves and returns the most appropriate response for a given post. Most of the features used in this model are latent semantic features, which can't capture the deep semantic relevance between posts and responses. In this paper we start from this dataset and combine shallow features and deep features to improve the performance of the short-text conversation.

Deep learning is another approach to solve this task. Such works in that thread include deep architecture for matching short texts (DeepMatch) proposed by Lu and Li [16] and deep structured semantic models for information retrieval proposed by

Huang et al. [8]. Lu and Li [16] use a deep architecture to combine the localness and hierarchy intrinsic to natural language problem by using a massive dataset. However, this architecture almost use the traditional features instead of deep features of short text (i.e., word embedding). Most of these works, which claimed to have used deep architecture, are embedding-based models. Word embedding, also called distributed representation of words [1, 6, 9, 15], shows strong power for measuring syntactic and semantic word similarities and has achieved success in many NLP tasks such as sentiment analysis [12] and statistical machine translation [7]. Our work is also related with Socher et al. [11], which use the unfolding recursive auto-encoder and parsing tree to construct the interaction matrix of two sentences. This strategy combining with dynamic pooling achieves state-of-the-art performance on paraphrase identification task.

3 Matching Short Texts Using Both Shallow and Deep Features

The previous work [3] learned a ranking model with 15 shallow matching features, which are learned from matching models or generated directly from post and response. In our method, we build three new matching models and learn both shallow features and deep features between posts and responses. Then we build a ranking model to evaluate the candidate responses with both shallow features and deep features.

For shallow matching features, we first introduce a variant of vector space model to measure post-response similarity by a word-by-word matching. Then we use LSI (Latent semantic indexing) model to capture the latent semantic matching features, which may not be well captured by a word-by-word matching. For deep matching features, we construct a matching models based on neural networks and word embedding.

3.1 Shallow Matching Features

Post-Response Similarity: To measuring the similarity between a post and a response by a word-by-word matching, we use a simple vector space model. For example, given a post and a response, the sparse representations of them in vector space model are x and y, then the Post-Response Similarity can be measured by the cosine similarity.

$$\mathrm{Sim}(x, y) = \mathrm{CosineSim}(x, y) = \frac{x^T y}{\|x\|\|y\|}$$

Unlike the traditional vector space model such as Bag-of-words model, in our model the values in the vector are the TF-IDF weights of words. Vector space model based on TF-IDF weights is valuable words bias and shows better performance than traditional Bag-of-words model in the experiment.

Post-Response Latent Semantic Similarity: LSI model has been used in many NLP task for measuring latent semantic matching. It learns a mapping from the original sparse representation to a low-dimensional and dense representation for text. For example, we represent post as vector x and response as vector y. Using LSI model, we map x and y to low-dimensional vector representations x_{lsi} and y_{lsi}. Similarly, the semantic similarity between the post and the response can be measured by the cosine similarity of x_{lsi} and y_{lsi}.

$$\mathrm{SemSim}_{PR}(x_{lsi}, y_{lsi}) = \mathrm{CosineSim}(x_{lsi}, y_{lsi})$$

Post-Post Latent Semantic Similarity: Inspired by previous work [3], we also consider the semantic similarity between a post and a post. The intuition is that a response y is suitable for a post x if its original post x' is similar to x. So we use semantic similarity between x and the original post x' to measure this kind of indirect correlation between a post and a response. Here we also use the LSI model to measure the post-post semantic similarity.

$$\mathrm{Cor}(x, y) = \mathrm{SemSim}_{PP}(x_{lsi}, x'_{lsi}) = \mathrm{CosineSim}(x_{lsi}, x'_{lsi})$$

3.2 Deep Matching Feature

As is stated above, we can acquire shallow matching features between a post and a response. To learn the deep matching features between posts and responses, we propose a matching model based on neural networks and word embedding.

With the learned word embeddings, we first map every word of posts and responses to a unique vector. As shown below, given a post x or a response y, we first convert them into a set of vectors.

$$x = (w_1\ w_2\ w_3\ \ldots\ w_i\ \ldots) \rightarrow X = (v_1\ v_2\ v_3\ \ldots\ v_i\ \ldots)$$

$$y = (w_1\ w_2\ w_3\ \ldots\ w_j\ \ldots) \rightarrow Y = (v_1\ v_2\ v_3\ \ldots\ v_j\ \ldots)$$

Here w_i is the i-th word in x and v_i is the corresponding word embedding. By measuring cosine similarity for each vector pair in X and Y, we can get a correlation matrix M_{cor}.

$$M_{cor} = \left(\mathrm{CosineSim}(v_i, v_j)\right)\ v_i \in X, v_j \in Y$$

The size of this correlation matrix is variable for variable-length posts and responses. In order to use neural networks and prevent the dimension of the feature space becoming too large, we need to convert variable-length posts and responses to fixed-length. We sort all the words in post x and response y by their TF-IDF weights in descending order, then we choose the top m words in x and top n words in y. Finally, we can get a correlation matrix with size $m*n$.

$$sorted(x) \xrightarrow{top\ m\ words} x' = (w_1\ w_2\ w_3\ \ldots\ w_m) \rightarrow X' = (v_1\ v_2\ v_3\ \ldots\ v_m)$$

$$sorted(y) \xrightarrow{top\ n\ words} y' = (w_1\ w_2\ w_3\ \ldots\ w_n) \rightarrow Y' = (v_1\ v_2\ v_3\ \ldots\ v_n)$$

$$M_{cor}{}' = \left(\text{CosineSim}(v_i, v_j)\right)\ v_i \in X', v_j \in Y'$$

Here *X'*, *Y'* and M_{cor}' have fixed size. For the post with a length less than *m* or the response with a length less than *n*, we set zero for the corresponding value in the correlation matrix. At the last step of the model, we flatten M_{cor}' to a feature vector with the same size *m*n*. Then we build neural networks with single hidden layer, which uses values of the feature vector as input features and outputs a matching score. The whole model is shown in Figure 1.

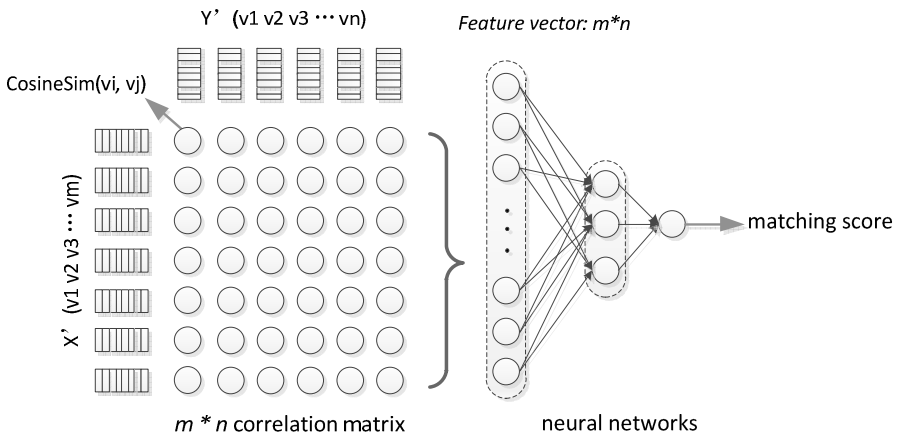

Fig. 1. Matching model based on neural networks and word embedding

Training: For training this model, we use the ranking-based strategy. Given a post *x*, the model should output a higher score for positive response *y+* than negative response *y-*. So our instance for training is (*x, y+, y-*). We use the unlabeled data to train our embedding matching model. For a post *x*, every original response of *x* is used as positive response *y+*, and choose a response randomly from the response dataset as negative response *y-* for each *y+*. Hence, we get the following ranking-based loss as objective:

$$E_\theta(x, y^+, y^-) = max\left(0, \alpha + s(x, y^-) - s(x, y^+)\right)$$

Where $s(x, y)$ is the output matching score for *(x, y)*, α is the margin between positive response and negative response, θ is the parameters for the embedding match model. The optimization is relatively straightforward with the back-propagation.

3.3 Other Matching Features

In addition to the matching features generated from the matching models above, we also use some handcraft features for this task, which can describe the relevance between post and response for some special cases.

- $Common\_Entity(x, y)$: This feature measures whether post x and response y have same entity words such as first and last names, geographic locations, addresses, companies and organization name. The intuition here is that a post and a response in the nature conversation usually contain some common key words.
- $Common\_Number(x, y)$: This feature measures whether post x and response y have same number such as date and money.
- $Common\_Words\_Lenght(x, y)$: This feature indicates the length of the longest common string between a post and a response. In social media such as Micro-blog, a response can be forwarded as a new post. We may find a response which is very similar to the given post, but it's not a suitable response. With this feature, we can filter this kind of responses.
- $Length\_Ratio(x, y)$: This feature indicates the ratio of the length of post x to the length of response y.

3.4 Ranking and Combination of Features

At the last step of our method, a ranking model with all the matching features above is learned to further evaluate the candidate responses. As shown in Figure 2, given a post, the ranking model gives a matching score for each candidate response. Then we pick the response with the highest matching score from the candidate set as suitable response for the post.

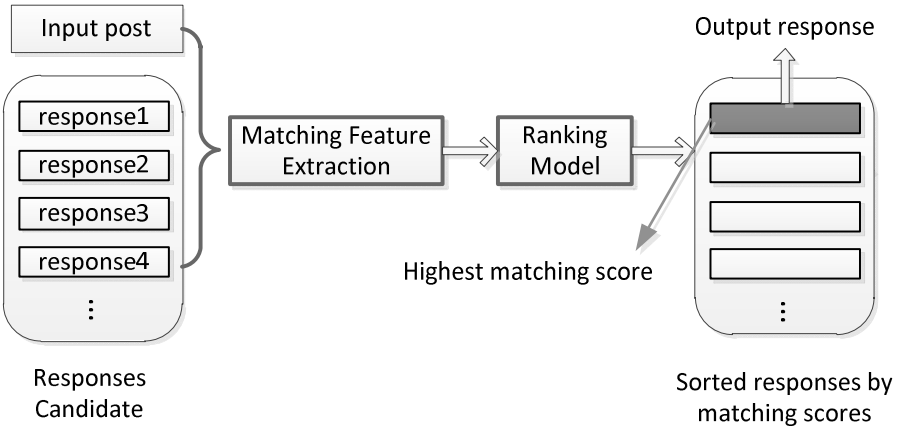

Fig. 2. Ranking model for short texts matching

The ranking function of the ranking model is defined as following, which is a linear score function and trained with RankSVM [14].

$$Score(x, y) = \sum_{i=1}^{n} w_i \Phi_i(x, y)$$

Here $\Phi_i(x, y)$ stands for the acquired matching features and w_i is the weight of $\Phi_i(x, y)$ to be learned.

4 Experiment

4.1 Dataset

In our experiments, we made use of the dataset of short-text conversation based on the real-world instances from Sina Weibo, which is published by Wang, et al. [3], as shown in Table 1.

Table 1. Dataset of short-text conversation based on the real-world instances from Sina Weibo

Dataset	Size
Unlabeled post-response pairs	38,016 posts, 618,104 responses
Labeled post-response pairs	422 posts, 12,402 responses
Vocabulary	125,817

This dataset includes two parts: unlabeled post-response pairs and labeled post-response pairs. The unlabeled post-response pairs are training set, which contains 38,016 posts and 618,104 responses. The labeled post-response pairs are test dataset, which contains 422 posts and 12,402 responses, and there are about 30 candidate responses for each post. The vocabulary of this dataset contains 125,817 words.

For each post-response pair, a post and a response are represented as *post_id* and *response_id*. In the labeled post-response pairs, each pair is labeled as a matched pair (marked as 2) or a mismatched pair (marked as 1) by human. An example of unlabeled post-response pair is shown below:

Post-response pair: 10:81,213

- *Post: 10##2012 年 来 了，祝 好运 、健康 、佳肴 伴你 度过 一个 快乐 新年 。*
- *Post: 10##2012 is coming. Good luck, good health, hood cheer. I wish you a happy New Year.*
- *Response 1: 81##林 老师， 新年 快乐 。*
- *Response 1: 81##Teacher Lin, happy new year!*
- *Response 2: 213##祝 教授， 工作顺利，身体健康！*
- *Response 2: 213##Happy new year, good health, professor.*

In order to train word embedding, we also build a Weibo dataset of 33,405,212 posts. We filter out the posts with length less than 5 and the meaningless posts in the dataset. Jieba[1], a famous open source software, is used for word segmentation. After preprocessing and word segmentation, we use the method introduced by Mikolov et al. [6] for learning word embedding. In our experiments, we use a

[1] https://github.com/fxsjy/jieba

particular implementation of this model[2] and the length of word embedding is set to 100. After training, we learned a set of word embeddings with a vocabulary of size 810,214.

4.2 Experiments and Benchmark

To training matching models in Section 3, we use unlabeled post-response pairs introduced in the dataset. For vector space model based on TF-IDF, the TF-IDF values for words can be generated directly from corpus. For training LSI model, we concatenate each post and its responses to get informative documents. For matching model based on word embedding, we need to sort all the words in a post and a response by their TF-IDF weights in descending order, and choose the top m words in the post and top n words in the response for generating matching features. By using comparison experiments, we can get best effect with 15 for m and 10 for n.

After training matching models and generating matching features, we train the ranking model defined in Equation 1 with the implementation of RankSVM[3]. Specifically we perform a 5-fold cross-validation on the labeled post-response pairs.

Our model will return the response with the highest matching score. So the evaluation of our models is based on the **P@1.** This measures the precision of the response returned by model.

$$\text{P@1} = \frac{Count(top\ 1\ response\ is\ matched)}{Count(posts)}$$

4.3 Performance

The results of experiments are shown in Table 2. Our competitor methods include retrieval-based model [3] and DeepMatch [16] model. For the DeepMatch model, we re-implement it and train it on the unlabeled training dataset. VSM stands for the matching features generated by vector space model based on TF-IDF. LSI stands for the matching features generated by LSI model. The deep feature is generated by embedding match model.

Table 2. Comparison of different features

Model and Features	P@1
Retrieval-based Response Model [3]	0.574
DeepMatch [16]	0.424
Shallow Features (VSM, LSI)	0.577
Shallow Features + Deep Features (VSM, LSI, Embedding Match)	0.612
All Features	**0.637**

[2] http://radimrehurek.com/2014/02/word2vec-tutorial/
[3] http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html

From the results, we can see that just using the word features generated by VSM model and latent semantic model, the result is as well as retrieval-based model. When combined with deep features, the performance significantly outperforms retrieval-based model and those just using shallow features. This demonstrates that although the shallow features are effective for short text semantic matching, it can't capture the deep semantic relevance information between two text object'. After adding other handcraft features the performance can be improved further, it implies that deep features cannot cover some relevance for some special cases, for example the "上海 天气"(The weather in Shanghai) and"海圳 天气"(The weather in Shenzhen) case. The main reason for the bad performance of DeepMatch may be that the training dataset is not big enough for this deep architecture.

5 Conclusion

In this paper, we design an embedding match model to generate deep features for short text matching and study the effect of shallow features and deep features on the performance. Experiments show that the deep features cover rich semantic relevance information between post and response, which the shallow features cannot capture. Nonetheless, experiment also shows that shallow features are necessary for some special cases of semantic matching. Combining the shallow features and deep feature generated by embedding match model, we get the state-of-the-art performance on the dataset released by Wang et al. [3].

Although the deep feature has proved significantly effective in this paper, there is still much matching information, which it cannot capture. For future work, we will try to use deep architecture and massive data to extract rich deep features of short text matching.

Acknowledgement. This paper is supported in part by grants: NSFCs (National Natural Science Foundation of China) (61173075 and 61272383), Strategic Emerging Industry Development Special Funds of Shenzhen (ZDSY20120613125401420 and JCYJ20120613151940045), Key Basic Research Foundation of Shenzhen (JC20100 5260118A and JC201005260175A) and Science and Technology Funds of Guizhou Province (黔科合J字[2013]2335).

References

1. Mnih, A., Hinton, G.: Three new graphical models for statistical language modelling. In: International Conference on Machine Learning, ICML (2007)
2. Leuski, A., Traum, D.R.: Npceditor: Creating virtual human dialogue using information retrieval techniques. AI Magazine 32(2), 42–56 (2011)
3. Wang, H., Lu, Z., Li, H., Chen, E.: A dataset for research on short-text conversations. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, pp. 935–945 (2013)

4. Williams, J.D., Young, S.: Partially observable markov decision processes for spoken dialog systems. Comput. Speech Lang. 21(2), 393–422 (2007)
5. Schatzmann, J., Weilhammer, K., Stuttle, M., Young, S.: A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. Knowl. Eng. Rev., 97–126 (2006)
6. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
7. Mikolov, T., Le, Q.V., Sutskever, I.: Exploiting similarities among languages for machine translation. CoRR, abs/1309.4168 (2013)
8. Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., Heck, L.: Learning deep structured semantic models for web search using clickthrough data. In: Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management, pp. 2333–2338. ACM (2013)
9. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research (JMLR) 12, 2493–2537 (2011)
10. Jafarpour, S., Burges, C.J.C.: Filter, rank, and transfer the knowledge: Learning to chat (2010)
11. Socher, R., Huang, E.H., Pennington, J., Ng, A.Y.: Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In: Advances in Neural Information Processing Systems (2011)
12. Socher, R., Huang, E.H., Pennington, J., Ng, A.Y., Manning, C.D.: Semisupervised recursive autoencoders for predicting sentiment distributions. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (2011)
13. Misu, T., Georgila, K., Leuski, A., Traum, D.: Reinforcement learning of question-answering dialogue policies for virtual museum guides. In: Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL 2012, pp. 84–93 (2012)
14. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002, pp. 133–142. ACM, New York (2002)
15. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. Journal of Machine Learning Research (JMLR) 3, 1137–1155 (2003)
16. Lu, Z., Li, H.: A deep architecture for matching short texts. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 26, pp. 1367–1375. Curran Associates, Inc. (2013)