Semantic Matching Using Deep Multi-Perception Semantic Matching Model with Stacking

Xu Cao[⋆], Xinran Liu[⋆], Binjun Zhu[⋆], Qingliang Miao, Changjian Hu, Feiyu Xu

Building H, No.6, West Shangdi Road, Haidian District Beijing, China
{caoxu3, liuxr7, zhubj4, miaoql1, hucj1, fxu}@lenovo.com

Abstract. Semantic matching is one of the most basic tasks of Natural Language Processing(NLP). It is of great importance for various NLP tasks such as automatic question answering, chat bot, information retrieval, and machine translation. This work is an integrated solution for CCKS 2018's Task 3 matching the question intent for the real customer service corpus in Chinese. The Deep Multi-Perception Semantic Matching (DMPSM) model is proposed for this task. In addition, we implement various machine learning and deep learning models. And the stacking technology is applied to model fusion and the final result prediction. DMPSM got the best performance, which reached the 0.838 F1 score in develop data set. After the model fusion, the ensemble learning method improved the F1 score to 0.855, which is a significantly enhanced evaluation.

Keywords: Semantic Equivalence Identification, Paraphrase Identification, Sentence Pair Matching, Deep Learning, Model Fusion

1 Introduction

Sentence Pair Matching is a common task in NLP domain. The challenging of the semantic matching is that the matching requirements are different, and the definition of matching can be also flexible. Existing semantic models, such as Word2Vec, LDA, etc., can determine the similarity between text. But it is difficult to say whether the two paragraphs of text are saying ”the same thing”. For example, when we try to solve the classic sentence retelling problem, it is necessary to judge whether the two sentences are merely different in expression, but the meaning is the same.

However, as Quora question-matching corpus is published, a large number of sentence matching research work carried out the definition of the corpus publisher, which is called semantic equivalence discrimination, semantic equivalence judgment, or equivalent, rather than directly judge whether the two sentences have the same semantics, the core problem turn into the intent match of the sentence.

The Task 3 of China Conference on Knowledge Graph and Semantic Computing (CCKS) 2018 is closer to the actual needs of NLP tasks such as smart customer service due to the data source is coming from real Q&A corpus. Given two sentences, the solution is supposed to determine whether the intentions are the same or similar. All corpora come from the real intelligent bank customer service log and have been screened and manually intent-matched.

[⋆] Equal contribute

In this paper, we not only implement various of deep semantic models, but also propose a new architecture to solve this problem, called DMPSM model. In detail, our model obtains multi-perception information, whole sentence's semantic meaning and words interaction feature, which help achieving the best score.

2 Related Work

2.1 Sentence Pair Modeling

Sentence pair modeling has attracted lots of attention in the past decades. Many tasks can be reduced to a semantic text matching problem. In the past decade, various models has been proposed to work on these tasks, and they can be divided into two types: sentence encoding models and sentence pair interaction models.

InferSent[1] uses row max pooling on hidden state sequence after Bi-directional LSTM (Bi-LSTM) to represent the semantic meaning of a sentence. Shortcut-stacked Sentence Encoder Model (SSE) based on the structure of InferSent. The additional shortcut connection between LSTM layers helps it to model deep semantic meaning avoiding training error accumulation. These two models represent sentence in a fixed vector and calculate the semantic relationship between sentences based on vector distance[2], belong to sentence encoding type. As for interaction models, the Decomposable Attention model (DecAtt)[3] is one of the earliest models to use attention-based alignment for sentence pair modeling. The Enhanced Sequential Inference Model (ESIM)[4] is closely related to the DecAtt model with some improvement. First, Chen et al[4] demonstrated that using Bi-LSTM to encode sequential contexts is important. Second, instead of using Multi-Layer Perception (MLP), ESIM also adapts the time-step model to model the information after alignment. MatchPyramid[5] model was proposed to model text matching as the problem of image recognition. It can successfully identify salient signals such as N-gram and N-term matching. MatchPyramid model is based on hard alignment which is different from the soft alignment used in DecAtt and ESIM. Bilateral Multi-Perspective Matching model (Bi-MPM) was proposed by Wang et al[6]. This model focuses on the bi-directional alignment and invents a new way to obtain multi-perspective cosine similarity which helps it achieve state-of-the-art performance.

2.2 Model Fusion

For each machine learning algorithm, the perspective to solve the problem is slightly different, which leads to different results they give for the same task. In this case, we introduce the concept of ensemble learning, or model fusion, to determine which result we would take as the final answer in an explicable way.

Voting Many techniques that aim to combine multiple results into a singular prediction are based on voting. Majority voting, or hard voting, consists in using the generated output from the different model and comparing them. The final decision about the classification is taking regarding the result with the majority votes. In other words, the minority is subject to the majority. In this work, we take the results of voting at first, and achieved better performance than singular method, this allows us to consider more alternatives idea of ensemble learning.

Stacking Another idea of model fusion is stacking, which can be considered as a method of meta-learning, or learning about learning. Stacking, or stacked generalization[7] is an approach for constructing classifier ensembles. The first step is to collect the output of each model into a new set of data. For each instance in the original training set, this dataset represents every model's prediction of that instance's class, along with its true classification. During this step, care is taken to ensure that the models are formed from a batch of training data that does not include the instance in question, in just the same way as ordinary cross-validation. The new data is treated as the data for another learning problem. In the second step, a learning algorithm is employed to solve this problem. K.M Ting's work[8] indicates that for successful stacked generalization it is necessary to use output class probabilities rather than class predictions. In our work, we combine multiple classifiers to induce a higher-level classifier with improved performance. The higher-level classifier can be thought of as the president of a committee with the ground-level classifiers as members. Ground-level classifiers often make different classification errors. Hence, a president that has successfully learned when to trust each of the members can improve overall performance.

3 Data Preprocessing and Feature Engineering

3.1 Data Preprocessing

Word Segmentation As we are supposed to deal with Chinese text, the problem of segmentation is unavoidable. Here we simply implement ”Jieba”[1] Chinese text segmentation, known as the best Python Chinese word segmentation module. We take accurate mode that attempts to cut the sentence into the most accurate segmentation, which is suitable for text analysis. Besides, we also add the segmentation of character-level in the end of each sentence, since the individual characters can also be deemed as words and involve abundant information in Chinese.

Synonyms and Typos In the process of data cleaning and checking, we find that there are appreciable quantity of synonymous and typos. Beyond that, simplified and traditional Chinese also appear blended. Therefore, we unify the expression standard and correct the typos while data preprocessing, to ensure the reliability of follow-up feature extraction.

Special Word Transform For some words with similar sense but in various form, we transform the words into special tag, like ”15”→”[dig]”, ”what”→”[inter_what]”. All the digit and 8 kinds of interrogative (how, what, when, why, where, who, quality, whether) in the text are transformed into the specific tag.

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

3.2 Feature Engineering

Statistical Features

Sentence Length Although normal sentences with similar meaning can have various length, however, in the real customer service records, people tend to use relatively brief description to state their issues. It is necessary to consider the how the length of two sentence are different. We extract the feature of sentence length by following formula:

$$\frac{|length\ of\ sentence_1 - length\ of\ sentence_2|}{2 \times (length\ of\ sentence_1 + length\ of\ sentence_2)} \tag{1}$$

Special question words One of anther distinguishing feature of these sentences is the utilization of special question words. Wh-questions in English have an extensive application, but with their varied structures, while as for Chinese, the words can indicate some kinds of question type very specifically. We give these special words identification codes and add them to the feature vector for each sentence.

BLEU Score with N-gram Bilingual Evaluation Understudy (BLEU)[9] is an evaluation score comparing the candidate text translation with one or more other reference translations proposed by K. Papineni in 2001. Although BLEU was originally developed for translation work, it can also be used to evaluate the quality of text generated for a set of natural language processing tasks. In our task, we consider BLEU as a way to comparing the similarity of the sentence pair. A perfect match has a score of 1, and a complete not match has a score of 0. This evaluation method counts the matching N-gram words or phrases in one sentence and another, a word (called 1-gram or uni-gram) compares each word, and a bi-gram will compare every word pair. This comparison is regardless of the order of the words.
We add the clipped N-gram counts for one sentence and divide by the number of candidate N-grams in the other sentence to compute a modified precision score p_n.

$$p_n = \frac{\sum_{Ngram \in Sentence\ 1} Count_{clip}(Ngram)}{\sum_{Ngram\prime \in Sentence\ 2} Count(Ngram\prime)} \tag{2}$$

and the final BLEU score is the average of p_n of N-gram

Overlap Words We also take the co-occurrence words into consideration by counting how many words are same and how many words are unique in each sentence pair. After eliminate the effects of synonymous and two kinds of Chinese expression by data preprocessing we mentioned before, we can gain more overlap words information.

N-gram Distance The items of N-gram in a word sequence of a sentence would be transformed into binary vector. For two sentences, the distance of N-gram vector pairs can reflect the similarity. The cosine distance and L1, L2 distance are applied to calculate the distance.

Fuzzy Wuzzy Matching Fuzzy Wuzzy matching is a python package to calculate the differences between sequences. The degree of closeness between two strings is measured using Levenshtein Distance, also known as edit distance. There are four popular types of fuzzy matching logic supported by Fuzzy Wuzzy package: **Ratio** (uses pure Levenshtein Distance based matching), **Partial Ratio** (matches based on best sub-strings), **Token Sort Ratio** (tokenizes the strings and sorts them alphabetically before matching), **Token Set Ratio** (tokenizes the strings and compared the intersection and remainder).

Text Features

Sentence Embedding Our sentence embedding is based on word embedding trained by Word2Vec[10], TF-IDF scores, in addition to other features we introduced above. We train the word vector using training data set provided by CCKS, and the sentence vector is calculated by summing up the product of each word's vector and its TF-IDF score. In the end, we add the feature list to the sentence vector as the final embedding of sentence.

LSI Embedding Latent Semantic Indexing (LSI) is an indexing and retrieval method that uses a mathematical technique called singular value decomposition (SVD) to identify patterns in the relationships between the terms and concepts contained in an unstructured collection of text. LSI is based on the principle that words that are used in the same contexts tend to have similar meanings. A key feature of LSI is its ability to extract the conceptual content of a body of text by establishing associations between those terms that occur in similar contexts. All the text corpus is used to train a LSI model, and each query text can be convert into an LSI embedding.

LDA Embedding Latent Dirichlet Allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics. This model is also been trained by all the text corpus. And the topics value of one query are regards as a feature vector.

Similarity Features

Cosine Distance Since we have already acquired the embedding of the sentence in each pair, we firstly compute the cosine distance as a preliminary estimate of their similarity. Although we have plenty of complicated method to determined whether the two sentences is semantic matching, the basic judgment of mathematical way can be also useful as a important feature.

L1, L2 Distance L1 Distance (Manhattan Distance), L2 Distance (Euclidean distance) are two metrics to calculate the distance between two points in Euclidean space. The Minkowski Distance is a metric in a normed vector space which can be considered as a generalization of both the L1 Distance and the L2 Distance. The Minkowski Distance is defined as:

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{3}$$

Minkowski distance is typically used with p being 1 or 2, which correspond to the L1 Distance and L2 Distance.

Edit Distance Edit distance is a way of quantifying how dissimilar two strings or words are to one another by counting the minimum number of operations required to transform one string into the other. Different definitions of an edit distance use different sets of string operations. In this work, we use the Levenshtein distance operations, which contains removal, insertion, or substitution of a character in the string[11].

4 Method

4.1 Traditional Machine Learning Model

Logistic Regression Logistic Regression (LR) is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

Random Forest Random Forests (RF)[12] is an ensemble learning method for classification and regression tasks. The model construct a multitude of decision trees at training time and output the class or regression value. The result is calculated by classes voting (classification) or mean prediction (regression) of the individual trees. The training algorithm for random forests applies bagging technique to escape over-fitting.

XGBoost XGBoost (XGB) [13] is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

LightGBM LightGBM (LGB) [14] is a gradient boosting framework that uses tree based learning algorithms developed by Microsoft. The framework is fast and was designed for distributed training. It supports large-scale datasets and training on the GPU. In many cases LightGBM has been found to be more accurate and faster than XGBoost, though this is problem dependent. By default LightGBM will train a Gradient Boosted Decision Tree (GBDT), but it also supports random forests, Dropouts meet Multiple Additive Regression Trees (DART), and Gradient Based One-Side Sampling (Goss).

4.2 Deep Learning Models

MPCNN Multi-Perspective Convolutional Neural Networks was proposed by Hua He et al.[15] in 2015. It is a model for comparing sentences that uses a multiplicity of perspectives. They first model each sentence using a convolutional neural network that extracts features at multiple levels of granularity and uses multiple types of pooling, and then compare our sentence representations at several granularities using multiple similarity metrics. Based on the original model, we made improvements specially aimed at this task. We use the pre-trained word vector instead of random initialization vector, and extract features from characters, words and text level. Particularly, we add an attention-based input layer between the embedding layer and the multi-perspective sentence modeling layer. The reason for this modification is that in the MPCNN model, the two sentences are processed independently of each other, and have no interaction until the full-connected layer, which will lose a lot of useful information. The attention-based layer fuses the word embedding matrix of two sentences, and the new ”word vector” has the information of both sentences, which can better represent their interrelationship.

Bi-LSTM Bidirectional Recurrent Neural Network (BRNN) was invented in 1997 by Schuster and Paliwal[16], The basic idea of BRNNs is to connect two hidden layers of opposite directions to the same output. By this structure, the output layer can get information from past and future states. Long short-term memory (LSTM) units (or blocks) are a building unit for layers of a RNN, proposed in 1997 by Sepp Hochreiter and Jrgen Schmidhuber[17] and improved in 2000 by Felix Gers' team[18]. The problem with unidirectional RNN or LSTM is that it is processing from left to right, which results in the latter words being more important than the previous words. It is not reasonable for the task of sentence comprehension since each word should be equal, hence the Bi-LSTM that combines the both-way directional results.

Bi-MPM Bilateral Multi-Perspective Matching model (Bi-MPM) was proposed by Wang et al[6]. It is a model which uses some sorts of word alignment mechanisms(e.g., attention) then aggregate inter-sentence interactions, essentially belongs to the ”matching-aggregation” framework. Bi-MPM focuses on the bi-direction matching of two sentences and defines a new function to calculate the similarity of vectors. As for the function, Wang's method can obtain meanings of similarity in different ways, called multiple perspectives cosine similarity.
Bi-MPM pays attention to the matching module which is used to obtain the similarity of the two sentence in time-steps. Moreover, the matching is bi-directional, means that the left sentence and the right sentence will matching each other and get the matching information from their respective.

InferSent&SSE Different from MPCNN and Bi-MPM InferSent[1] and SSE[19] belong to framework ”Sentence encoding model” framework. Sentence encoding models focus on learning vector representations of individual sentences and then calculate the semantic relationship between sentences based on vector distance. InferSent model is invented by Facebook and is a model trained on the Stanford Natural Language Inference (SNLI) data set for transfer learning. They find that an encoder based on a Bi-LSTM architecture with max pooling is the best way to obtain the semantic information of a sentence. SSE model follows the work of InferSent. The encoder of SSE model is based on stacked Bi-LSTM with shortcut connections and fine-tuning of word embeddings (we make it fixed). Shortcut connection of LSTM layer can obtain deep semantic meaning, void training error accumulation and also help to improve the re-flow of gradient.

4.3 Deep Multi-Perception Semantic Matching Model

The existing model is either concerned about the information of encoding, or the interaction between words. In this case, we propose a new architecture that can sense the overall semantics of a sentence pair without neglecting the matching information between words called Deep Multi-Perception Semantic Matching (DMPSM) model. Fig. 1 shows the detail of the combined model. DMPSM achieves better performance due to the multi-perception module. We use row maxpooling[1] to capture the specific semantic meaning of the two sentences as $\vec{u}$, $\vec{v}$. Abs($\vec{u}$ - $\vec{v}$) and $\vec{u}$ * $\vec{v}$ are performed to represent the semantic distance. Moreover, interaction information is calculated by multi-perspective cosine similarity as [6]. These two perception of semantic information will be concated for aggregating.

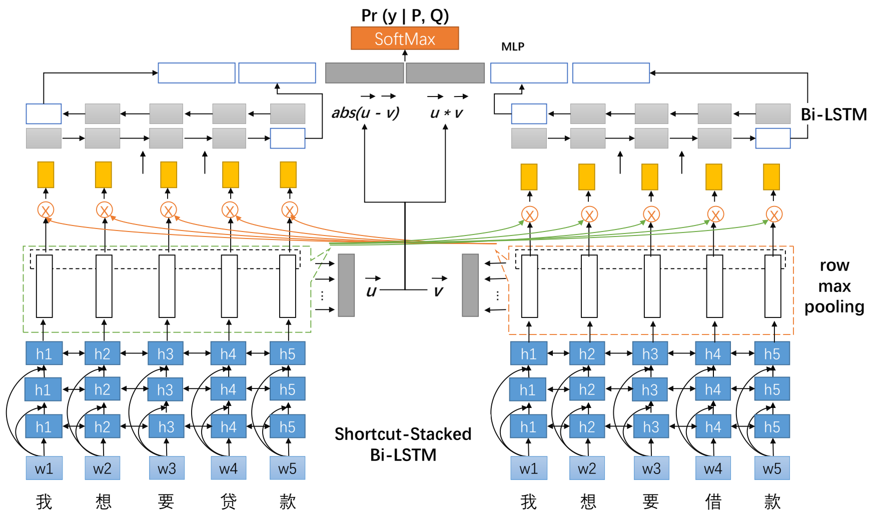

Fig. 1. Deep Multi-Perception Semantic Matching Model

5 Experiment and Result

5.1 Data set

We propose a multi-model fusion approach to this semantic matching task and experimented on standard data set provided by CCKS, which includes 100,000 pairs of training sentences and 10,000 pairs of development data set, and finally we test our approach on a 100,000 pairs of test sentences. Each pair of data contains two Chinese sentences which were selected from the real customer service record by CCKS. In addition to sentence pair, training data give each pair of data a 0/1 label to classify these sentences are semantic matching or not.

5.2 Training Details and Implementation

Data Partition Five-fold cross-validation is preformed on this data set. In each fold, one part is held-out for validation, another part is held-out for testing and the learning algorithm is trained on the remaining data. (see Fig. 2)

Parameters (1) Word Vector (Fixed in Training): We use open source Gensim toolkit[2] and set the embedding dimension is 100, mincount is 3. Moreover, the vector is trained on word & char data set for multi-granularity information. (2) Bi-LSTM: This layer is commonly used in our deep models such as Bi-LSTM, InferSent, SSE and Bi-MPM, we think 100 dim is suitable. (3) Dropout: It is important to use dropout in deep model, in Bi-MPM dropout rate is setted at 0.1 for better result. (4) Highway: In order to make the training process faster and avoid gradient vanishing, highway network are used as a trick for training deep models. We add this layer on both embedding layer and aggregate layer in Bi-MPM & DMPSM.

Stacking We conduct sentence pair modeling experiments on all the models showed on Fig1 level 0. For stacking, the models in level 1 are trained by the outputs of the models in level 0 combined with its real labels. Every models' (level 0) prediction on test set will be a feature to the models which we have trained in level 1. So after stacking, we will get three results predicted by LR, XGB and MLP and the final result is the average of these three models.

5.3 Result & Discussion

Because of the limited submission number, we only submitted and saved some deep model with better results. Other simple models with synthetic feature like Logistic Regression, Random Forest, XGBoost, LightGBM and also deep model InferSent are used in stacking to get better performance. Table 1 illustrates the results of different approaches.The analysis focus on two perspectives and it will help us find the best single model architecture.

[2] https://github.com/RaRe-Technologies/gensim

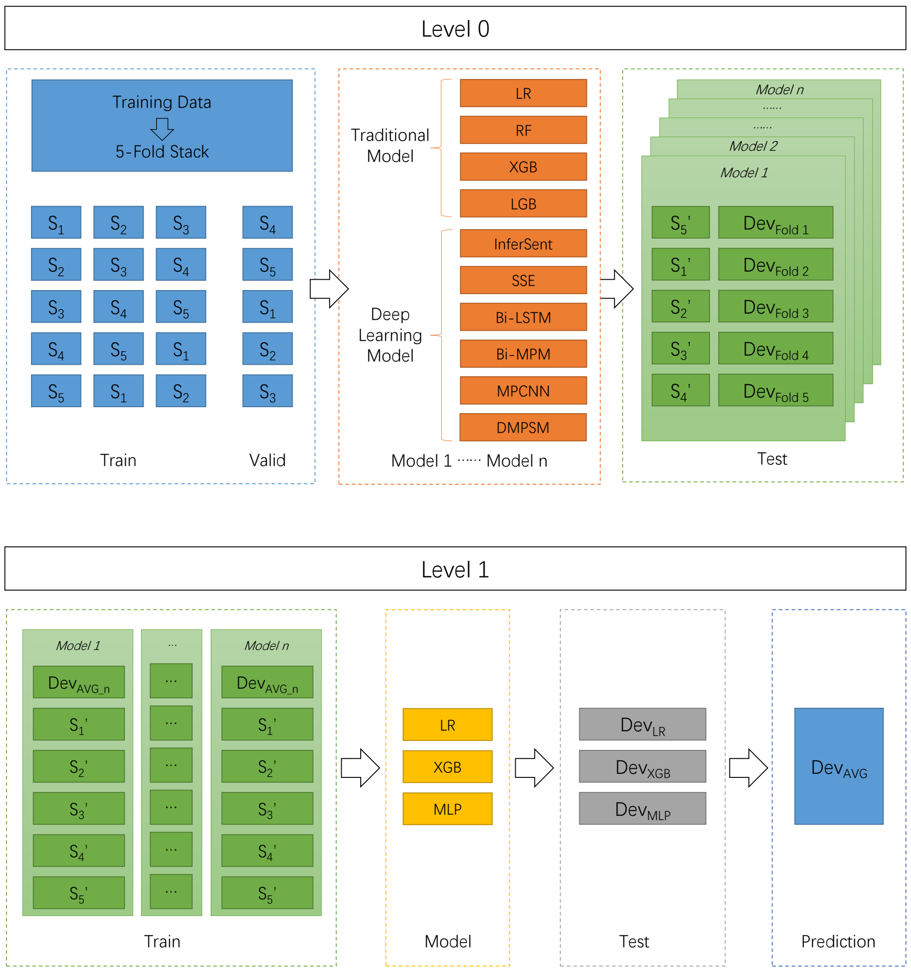

Fig. 2. Stacking

Table 1. Result of Different Approaches (Dev Set)

Model	F1	Precision	Accuracy	Recall
MPCNN	0.7882	0.7357	0.7719	**0.8488**
Bi-LSTM	0.8209	0.8236	0.8215	0.8182
Bi-MPM	0.8244	0.8242	0.8244	0.8246
SSE	0.8187	0.8252	0.8202	0.8124
DMPSM	**0.8387**	**0.8708**	**0.8445**	0.809
Stacking	**0.8559**	0.8288	**0.851**	**0.8848**

Which kinds of model can better obtain the semantic meaning of word sequence?CNN can conveniently captures N-gram feature by multi-size kernels which will help matching sentence in different granularities. It looks at a sentence from a visual point of view, and ignores the sequence relationship of words to some extent. In contrast, LSTM is often used for language model, and good at modeling sequence information. We can find that MPCNN get lower score than Bi-MPM Bi-LSTM or SSE, and as lan et al[2] indicates that encoding sequential context information with LSTM is critical.

Which kinds of model is better, Sentence encoding model or Interaction model? SSE model score is slightly lower than Bi-MPM, and it can be seen from the lan et al[2]'s analysis that the interactive model is generally better than encoding based model. However, we can ignore which one is better. These two types of models solve problems from different angles, we combine the two ideas and propose the DMPSM model.

For better performance, we perform stacking on these deep models with some simple models (LR, RF, XGB, LGB). Experiments show that setting the threshold to 0.25 can balanced the 0/1 results and gives the best results.

6 Conclusion and Future Work

Matching natural language text semantically is a key task for many applications in NLP. Different from semantic matching tasks based on common public data sets, our work pays more attention to the processing and understanding of real customer service data. With the in-depth study, we attempt multiple perspectives and ideas, which makes our research more practical and theoretical supported. Meanwhile, the implement of model fusion technique blends the viewpoint of different methods, from the experimental results, the method that this thesis puts forward comes to the result of our expectation. Despite our many attempts and efforts, there is still room for improvement in accuracy. Because of restrictions on data usage rules, we only train the word vector base on given training corpus, which led to the lacking of prior knowledge of professional field. In the future work, the introduction of knowledge graph of special domain may enhance the training efficiency. Furthermore, the other idea is taking advantage of model or knowledge from other areas and making more reasonable usage of these scarce data by transfer learning.

References

1. Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.
2. Wuwei Lan and Wei Xu. Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. In *Proceedings of COLING 2018*, 2018.
3. Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.
4. Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.
5. Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. Text matching as image recognition. . *In Proceedings of the thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, 2016.
6. Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017.
7. David H Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.
8. Ming Ting Kai and Ian H. Witten. Stacked generalization: when does it work? In *Fifteenth International Joint Conference on Artifical Intelligence*, pages 866–871, 1997.
9. Papineni, Kishore, Roukos, Salim, Ward, Todd, Zhu, and WeiJing. Bleu: a method for automatic evaluation of machine translation. *Acl*, 4(4):307–318, 2001.
10. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Computer Science*, 2013.
11. Gonzalo Navarro. A guided tour to approximate string matching. *Acm Computing Surveys*, 33(1):31–88, 2001.
12. Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.
13. Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.
14. Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.
15. Hua He, Kevin Gimpel, and Jimmy Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586, 2015.
16. Mike Schuster and Kuldip K Paliwal. *Bidirectional recurrent neural networks*. IEEE Press, 1997.
17. Alex Graves. *Long Short-Term Memory*. Springer Berlin Heidelberg, 2012.
18. Felix A. Gers, Jrgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. neural computation 12(10): 2451-2471. *Neural Computation*, 12(10):2451–2471, 2000.
19. Yixin Nie and Mohit Bansal. Shortcut-stacked sentence encoders for multi-domain inference. *arXiv.org, pp. arXiv:1708.02312*, 2017.