A Hybird System for Clinical Semantic Textual Similarity

Ying Xiong[1], Shuai Chen[1], Yedan Shen[1], Xiaolong Wang, PhD[1], Qingcai Chen, PhD[1], Jun Yan PhD[1], Buzhou Tang, PhD[1*]

[1]Key Laboratory of Network Oriented Intelligent Computation, Harbin Institute of Technology, Shenzhen, GuangDongGuangdong, China

[*]Corresponding author: tangbuzhou@gmail.com

Abstract

Semantic Textual Similarity (STS) that measures semantic similarity between text snippets can be used in various NLP applications such as summarization, question answering (QA), etc. A shared task regarding clinical STS was launched in 2018 (OHNLP 2018). The goal of this shared task is to reduce the cognitive burden in clinical decision-making process for providers. We developed systems based on Attention-Based Convolutional Neural Network(ABCNN) and ABCNN combined Bi-direction Long Short Term Memory networks(Bi-LSTM) for the shared task, with a highest Pearson correlation of 0.8143.

Keywords

clinical semantic textural similarity; ABCNN; Bi-LSTM; Pearson correlation

1.Introduction

The widespread adoption of Electronic Health Records (EHRs) provides a way to electronically record the patient's medical condition, thoughts, and behaviors in the medical team. EHRs helps the medical team make decisions. However, this also brings new challenges. A large amount of copy-and-paste and redundant data imposes a certain burden on medical decision makers. Semantic Textual Similarity (STS) that computes semantic similarity between text snippets is a good choice to minimize redundant information. STS is an indiepensible part of natural language understanding (NLU), and has been widely applied to various NLP applications such as textual entailment, information retrieval, paraphrase identification, plagiarism detection and sentence pairs judging[1]. In the general English domain, the SemEval Semantic Textual Similarity (STS) shared tasks have been organized since 2012 to develop automated methods for the task [1][2][3]. However, in the clinical domain, there is no existing resource for STS.

In particular, the task of STS is to determine the degree of semantic equivalence ranging from 0 to 5, with 0 meaning unrelated and 5 semantically equivalent[4][5]. For example, the sentence pair “The boy is playing basketball” and “The young man likes watching football matches” is scored as only 1 and the pair “How old are you” and “What is your age” is given a score of 5.

There are three typical approaches to compute semantic similarity. The first one is the BOW model which treats each sentence as a bag of words and represents it with a vector[6]. The similarity between a pair of sentences is the similarity of the corresponding vectors such as cosine. The second approach uses corpus-based and knowledge-based measures of similarity to measure the semantic similarity of sentences[7]. The third approach is based on machine learning models that combine different measures and features such as lexical, semantic and syntactic[8]. In recent years, researchers have proposed a variety of neural network architectures[9][10][11] for STS.

In this paper, we put forward a hybrid model to do the semantic similarity task in the work of predecessors, and combine ABCNN[12] and Siamese Bi-LSTM to get a better and more stable model.

2. Material and Methods

Fig.1 shows an overview of our hybrid semantic textual similarity system for the OHNLP 2018 task2. It is a hybrid system based on Bi-direction Long Short Term Memory networks(LSTM) and a previous work which proposed by (Yin et al., 2015) called Attention-Based Convolutional Neural Network(**ABCNN)**[12]. In this system, features can be divided into two modules: deep learning module and feature engineer. Traditional features are extracted firstly, and deep learning features will train in a deep learning neural network. Deep learning module consists of ABCNN and Bi-LSTM. The detailed description of the system is presented below.

2.1 Dataset

In the OHNLP 2018 challenge, the organizer manually annotated 750 sentence pairs, we divided them into two parts: (1) 600 sentence pairs used as a training set; and (2) the remaining 150 sentence pairs used as a validation set.

2.2 Traditional features

The following features are extracted as the traditional NLP features. Before extracting features, a data preprocessing process will be executed first. Because there are a lot of digital descriptions in clinical text such as dose, time, etc. So all digital number will be replaced with English number, for example “24” is translated into “twenty-four”. Other process includes removing stop words and lemmatizing each word.

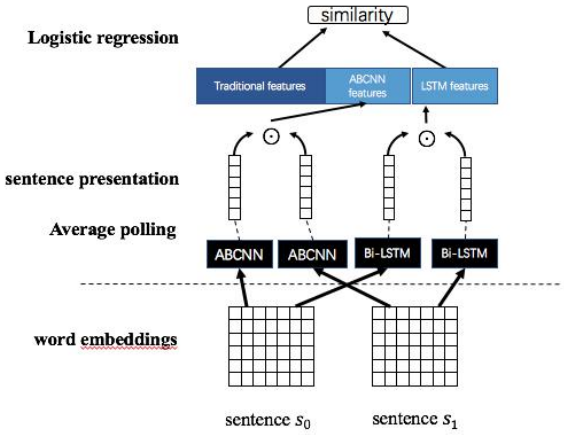

Fig.1 overview of our STS system for OHNLP2018

- Inverse Document Frequency(IDF) Features: IDF is a measure of how much information the word provides, that is whether the term is common or rare across all documents. It is an inverse of the logarithmical scale of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

$$idf(t,D) = log\frac{N}{|\{d\in D:t\in d\}|} \quad (1)$$

Where N is the total number of documents in the corpus, and in this dataset, N is the total number of all sentences in training set, $N = |D|$. $|\{d \in D:t \in d\}|$ is the number of documents where the term t appears.

- Sentence Length Features: The length of each sentence.
- N-gram Overlap Features: In order to capture the matching features of sentence pairs, S_i stands for the sets of consecutive n-grams, then the n-gram overlap(NGO) is defined as[8]:

$$NGO(S_1, S_2) = 2 \cdot \left(\frac{|S_1|}{|S_1 \cap S_2|} + \frac{|S_2|}{|S_1 \cap S_2|}\right)^{-1} \quad (2)$$

We calculated all 1,2,3 gram overlap on word-level and 2,3,4,5 gram overlap on character-level.

- Sentence representation features: Each sentence is represented by the average pooling of all word embeddings which is produced by word2vec (https://code.google.com/p/word2vec/). Table 1 lists two pretrained word embeddings in our work.

Table 1: three pretrained word embedding

Dimension	Source
200d	Wikipedia-pubmed-and-PMC-w2v
300d	GoogleNews-vectors-negative.bin

- Distance features and Similarity features: Inspired by Junfeng[13], it is irrational to concatenate the sentence representation directly mentioned above, because the dimension(1000d) is much larger than all of other traditional features' dimension(9d). Concatenating sentence representation and other traditional features directly would inhibit the representation of other traditional features. Therefore, calculating the similarity and the distance between the pairs of sentences are more effective. All similarities and distances are listed in Table 2.

Table 2: Similarity and Distance function

Type	Measures
Distance and Similarity	cosine distance
	Manhattan distance
	Euclidean distance
	Edit distance
	Chebyshev distance
	Manhattan similarity

Finally, all 26 traditional features are scaled into the same semantic space.

2.3 ABCNN features

ABCNN[12] was proposed by Yin et al. in 2015. They did their work based on a Siamese CNN[14], a twin structure that was proposed in 1993, that is, a model with two CNNs, each of which handles a sentence in a sentence pair, but the two CNNs share weights. Yin et al. [12] called this architecture BCNN. BCNN has four types of layers: input layer, convolution layer, average pooling layer and output layer. The input layer converts the input sentence into a word vector after padding. The convolutional layer use the convolution method proposed in Yoon Kim's paper[15], that is, a window only generates a convolution value at the end, and then slides over the length of the sentence to obtain a vector of length $length\_len + w_s - 1$ (w_s is wide Conv). In the pooling layer, the paper mentions two pooling layers, one is the last pooling layer "all-ap", and the other is the pooling layer used by the intermediate convolution layer "w-ap". The difference is that the window size is different when pooling. The output layer takes all the features obtained by the pooling layer as input, and then operates according to specific tasks. The purpose of this is to get different levels of abstract information. Based on BCNN, they proposed another three structures, ABCNN1, ABCNN2 and ABCNN3. These three structures are based on the BCNN to add attention mechanism. ABCNN1 adds the attention mechanism to the input layer in order to get the attention feature map to guide the convolutional layer to learn "counterpart-biased" sentence representation. ABCNN2 adds the attention mechanism to the output of the convolutional layer in order to improve the pooling result by weighting the convolution output result, so that the different words in the obtained high-level abstract features (short phrase, long phrase and so on) are superimposed according to different weights. Moreover, ABCNN2 adds fewer parameters and is less prone to overfitting. ABCNN3 combined with the above two to get a new model. ABCNN features in our methods are obtained from the pooling layer mentioned above.

2.4 LSTM features

Mueller and Thyagarajan present a Siamese adoption of the Long Short-Term Memory (LSTM) network for labeled data comprised of pairs of variable-length sequences[16]. Based on the previous work, we proposed our model. A basic LSTM containing following components: input gate(i_t), output gate(o_t), forget gate(f_t)， candidate value($\widehat{c_t}$), hidden state(h_t) and cell memory(c_t).A basic LSTM can be defined as:

$$
\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
\widehat{c_t} &= \tanh(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= i_t \odot \widehat{c_t} + f_t \odot c_{t-1} \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned} \tag{3}
$$

where $W_f, W_i, W_o, W_c, U_f, U_i, U_o, U_c$ are weights matrices， b_f, b_i, b_o, b_c are bias vectors and x_t is input vectors .

Information can be stored through the gates in LSTM, also, LSTM solve the problem of gradient vanishing. However, basic LSTM cannot capture the context information of sentence, but in natural language processing, context has a strong correlation, so we used Bi-LSTM instead. Bidirectional LSTM includes forward LSTM and reverse LSTM, forward LSTM is the basic LSTM, and reverse LSTM defined as:

$$
\begin{aligned}
f_t &= \sigma(\overleftarrow{W}_f x_t + \overleftarrow{U}_f \overleftarrow{h}_{t+1} + \overleftarrow{b}_f) \\
i_t &= \sigma(\overleftarrow{W}_i x_t + \overleftarrow{U}_i \overleftarrow{h}_{t+1} + \overleftarrow{b}_i) \\
o_t &= \sigma(\overleftarrow{W}_o x_t + \overleftarrow{U}_o \overleftarrow{h}_{t+1} + \overleftarrow{b}_o) \\
\widehat{c_t} &= tanh(\overleftarrow{W}_c x_t + \overleftarrow{U}_c \overleftarrow{h}_{t+1} + \overleftarrow{b}_c) \\
c_t &= i_t \odot \widehat{c_t} + f_t \odot c_{t+1} \\
\overleftarrow{h}_t &= o_t \odot \tanh(c_t)
\end{aligned} \tag{4}
$$

where $\overleftarrow{W}_f, \overleftarrow{W}_i, \overleftarrow{W}_o, \overleftarrow{W}_c, \overleftarrow{U}_f, \overleftarrow{U}_i, \overleftarrow{U}_o, \overleftarrow{U}_c$ are weights matrices， $\overleftarrow{b}_f, \overleftarrow{b}_i, \overleftarrow{b}_o, \overleftarrow{b}_c$ are bias vectors and x_t is input vectors .

On sentence pair representation $g1$ and $g2$, we calculated the cosine similarity between the two sentences. We used cosine similarity rather than Manhattan distance as our LSTM features because Manhattan distance did not work well.

2.5 Hybrid model

Because CNN has strong local representation ability for text, RNN has strong ability to represent the whole sentence. Therefore, we combined the advantages of both to obtain a hybrid model ABCNN-LSTM, which can get both the local representation of the sentence and the whole sentence. This model is also the model shown in Figure 1.

2.6 Evaluation

All of our results are tested on a separate test set. The test set consisted of 318 sentence pairs. All models use the Pearson correlation coefficient to analyze the sentence similarity. The Pearson correlation coefficient is defined as follows:

$$
r = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \tag{5}
$$

where n is the total number of samples, x_i and y_i are a sample with index i, $\bar{x}$ and $\bar{y}$ are the two samples' respective means.

2.7 Regression algorithms

We used a traditional machine learning algorithm and a fully connected layer of deep learning to make regression predictions. In machine learning, three learning algorithms were used for regression, they are Random Forest(RF), Gradient Boosting(GB), and XGBoost(XGB). RF and GB used scikit-learn tool(http://scikit-learn.org/stable/), and xgboost used xgboost tool(https://github.com/dmlc/xgboost). The average result of GB, RF and XGB was used as our final result. We reffered this kind of result as ML. For convenience, we referred to the full connection layer prediction as DL.

3. Result

All our experiment results on test set were listed in the table 3. We combined four kinds of models on the three types of features: the first was the combination of traditional NLP features and ABCNN features as the input of the logistic regression layer, and the output layer uses the machine learning regression and the network's fully connected layer to predict. When using machine learning regression, there are BCNN-ML, ABCNN1-ML, ABCNN2-ML and ABCNN3-ML in ABCNN model. ABCNN-Ensemble was the average of the above four methods. When using the fully connected layer of the network for prediction, the obtained models are abbreviated as BCNN-DL, ABCNN1-DL, ABCNN2-DL and ABCNN3-DL, respectively. The second was the combination of traditional features and Bi-LSTM features as the input to the final logistic regression layer. There were two results: LSTM-ML and LSTM-DL. The third was a hybrid model of ABCNN and Bi-LSTM. When using machine learning, the obtained models were abbreviated as BCNN-LSTM-ML, ABCNN1-LSTM-ML, ABCNN2-LSTM-ML and ABCNN3-LSTM-ML. When using network full connection layer prediction, the models obtained were BCNN-LSTM-DL, ABCNN1-LSTM-DL, ABCNN2-LSTM-DL and ABCNN3-LSTM-DL. ABCNN-LSTM-Ensemble was the average of BCNN-LSTM-ML, ABCNN1-LSTM-ML, ABCNN2-LSTM-ML and ABCNN3-LSTM-ML.

4. Discussion

From our best-performing results, we compared the distribution of similarity scores, shown in Fig.2. We also calculated the average MSE for each interval to compare our predicted performance in each interval, shown in Fig.3. We found the score distribution of the training set is similar to the score distribution of the test set, but our prediction results are poor in low scores and high scores, and most of them are concentrated in the middle score segments.

5. Conclusion

In the clinical text semantic similarity task, we used models currently performing well on text matching, including Siamese CNN and Siamese RNN. Performances of the models are unstable on the task as the corpus is too small.

Table 3: Results of all kinds of modelss

Model Name	Pearson R
BCNN-ML	0.7892
ABCNN1-ML	0.7874
ABCNN2-ML	0.7934
ABCNN3-ML	0.7867
LSTM-ML	0.7856
BCNN-LSTM-ML	0.8043
ABCNN1-LSTM-ML	0.7887
ABCNN2-LSTM-ML	0.8131
ABCNN3-LSTM-ML	0.8011
ABCNN-Ensemble	0.7925
ABCNN-LSTM-Ensemble	0.8121
All-Ensemble	**0.8143**
BCNN-DL	0.7956
ABCNN1-DL	0.7872
ABCNN2-DL	0.8053
ABCNN3-DL	0.7892
LSTM-DL	0.7955
BCNN-LSTM-DL	0.7829
ABCNN1-LSTM-DL	0.7818
ABCNN2-LSTM-DL	0.8090
ABCNN3-LSTM-DL	0.8041

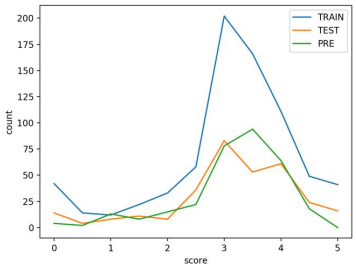

Fig.2 The similarity score distribution of training set, test set and our prediction

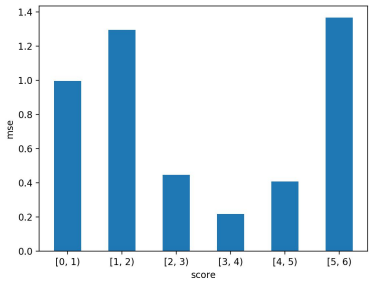

Fig.3 average MSE for each interval

Finally, we combined the results of different models and obtained better performance, with the best Pearson correlation coefficient of 0.8143. Our future work will consider extracting the entities in the text.

6. References

[1] E. Agirre *et al.*, "Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation," in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 2016, pp. 497–511.

[2] N. Afzal, Y. Wang, and H. Liu, "MayoNLP at SemEval-2016 Task 1: Semantic textual similarity based on lexical semantic net and deep learning semantic model," in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 2016, pp. 674–679.

[3] E. Agirre *et al.*, "SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability," in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Denver, Colorado, 2015, pp. 252–263.

[4] E. Agirre, M. Diab, D. Cer, and A. Gonzalez-Agirre, "Semeval-2012 task 6: A pilot on semantic textual similarity," in *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on* Semantic Evaluation, 2012, pp. 385–393.

[5] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "SemEval-2017 Task 1: Semantic Textual Similarity-Multilingual and Cross-lingual Focused Evaluation," ArXiv Prepr. ArXiv170800055, 2017.

[6] C. T. Meadow, B. R. Boyce, and D. H. Kraft, *Text information retrieval systems*, vol. 20. Academic Press San Diego, CA, 1992.

[7] R. Mihalcea, C. Corley, and C. Strapparava, "Corpus-based and knowledge-based measures of text semantic similarity," in *AAAI*, 2006, vol. 6, pp. 775–780.

[8] F. Šarić, G. Glavaš, M. Karan, J. Šnajder, and B. D. Bašić, "Takelab: Systems for measuring semantic text similarity," in *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, 2012, pp. 441–448.

[9] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

[10] R. Kiros *et al.*, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.

[11] D. S. Prijatelj, J. Ventura, and J. Kalita, "Neural Networks for Semantic Textual Similarity," in *International Conference on Natural Language Processing*, 2017.

[12] W. Yin, H. Schütze, B. Xiang, and B. Zhou, "Abcnn: Attention-based convolutional neural network for modeling sentence pairs," *ArXiv Prepr. ArXiv151205193*, 2015.

[13] J. Tian, Z. Zhou, M. Lan, and Y. Wu, "ECNU at SemEval-2017 Task 1: Leverage Kernel-based Traditional NLP features and Neural Networks to Build a Universal Model for Multilingual and Cross-lingual Semantic Textual Similarity," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017, pp. 191–197.

[14] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, 1994, pp. 737–744.

[15] Y.Kim "Convolutional Neural Networks for Sentence Classification," ,2014.

[16] J. Mueller and A. Thyagarajan, "Siamese Recurrent Architectures for Learning Sentence Similarity.," in *AAAI*, 2016, vol. 16, pp. 2786–2792.