Deep Feature Fusion Model for Sentence Semantic Matching

Xu Zhang[1], Wenpeng Lu[1, *], Fangfang Li[2, 3], Xueping Peng[3] and Ruoyu Zhang[1]

Abstract: Sentence semantic matching (SSM) is a fundamental research in solving natural language processing tasks such as question answering and machine translation. The latest SSM research benefits from deep learning techniques by incorporating attention mechanism to semantically match given sentences. However, how to fully capture the semantic context without losing significant features for sentence encoding is still a challenge. To address this challenge, we propose a deep feature fusion model and integrate it into the most popular deep learning architecture for sentence matching task. The integrated architecture mainly consists of embedding layer, deep feature fusion layer, matching layer and prediction layer. In addition, we also compare the commonly used loss function, and propose a novel hybrid loss function integrating MSE and cross entropy together, considering confidence interval and threshold setting to preserve the indistinguishable instances in training process. To evaluate our model performance, we experiment on two real world public data sets: LCQMC and Quora. The experiment results demonstrate that our model outperforms the most existing advanced deep learning models for sentence matching, benefited from our enhanced loss function and deep feature fusion model for capturing semantic context.

Keywords: Natural language processing, semantic matching, deep learning.

1 Introduction

Sentence semantic matching (SSM) is a fundamental research in many natural language processing tasks, such as Natural Language Inference [Mueller and Thyagarajan (2016); Liu, Sun, Lin et al. (2016); Wang, Hamza and Florian (2017); Gong, Luo and Zhang (2017)], Question Answering(QA) [Qiu and Huang (2015); Tan, Santos, Xiang et al. (2015); Zhang, Zhang, Wang et al. (2017)] and Machine Translation [Bahdanau, Cho and Bengio (2014)]. For example, a frequently answered question (FAQ) based QA system often organises question-answer pairs into tuples $(\mathrm{q}_i, \mathrm{a}_i)$ first, then try to figure out which question in question-answer pairs is the most semantically similar question to the given query sentence by SSM algorithms. Say $(\mathrm{q}_i, \mathrm{a}_i)$ is the optimal matched question-answer pair, the answer sentence a will be the target answer sentence for the FAQ based

[1] School of Computer Science and Technology, QiLu University of Technology (Shandong Academy of Sciences), Jinan, 250353, China.

[2] oOh! Media, Sydney, NSW, 2060, Australia.

[3] Centre of Artificial Intelligence, University of Technology Sydney, Sydney, NSW, 2007, Australia.

[*] Corresponding Author: Wenpeng Lu. Email: lwp@qlu.edu.cn.

QA system.

Last decade has been witnessing an amazingly increasing development of deep learning, many academic and industry giants have put a lot of effort for innovative deep learning models and applications. No doubt many NLP research tasks like SSM benefit a lot from innovative deep learning models as well. In deep learning based NLP direction, most research focus on sentence encoding to have better sentence feature vectors and better feature interaction matching [Kim (2014); Mou, Peng, Li et al. (2015); Mueller and Thyagarajan (2016)]. Let us take the most two popular deep learning model sets Convolutional neural networks (CNNs) and Recurrent neural networks (RNNs) as examples to detail a bit further. CNNs have been widely applied in QA [Qiu and Huang (2015); Zhang, Zhang, Wang et al. (2017)]] and text classification tasks [Kim (2014); Conneau, Schwenk, Barrault et al. (2016)]. CNN is very advantageous on extracting sequence features and sentence encoding [Blunsom, Grefenstette and Kalchbrenner (2014); Hu, Lu, Li et al. (2014); Yin and Schütze (2015); Zhang, Zhang, Wang et al. (2017); Bai, Kolter and Koltun (2018)]. RNNs are time-aware sequential deep learning model set, being able to transmit neural unit values from previous time states to the current neural units to optimise feature weights. RNN's memory function is advantageous for handing contextual sequential data. Obviously, textual sentence can be considered as words sequence, where RNNs can be easily applied to address textual tasks. In short, RNNs are excellent for sentence encoding through sequential data modelling [Hochreiter and Schmidhuber (1997); Gers and Schmidhuber (2000); Cho, Van Merriënboer, Gulcehre et al. (2014); Jozefowicz, Zaremba and Sutskever (2015); Tan, Santos, Xiang et al. (2015); Greff, Srivastava, Koutník et al. (2017)]. However, both CNNs and RNNs models cannot fully capture all features in the feature extraction or encoding process. This feature loss problem will further cause semantic context loss in understanding language semantics for very long sentences.

In order to solve the semantic information loss problem, and to remember the key semantic context much better in understanding textual sentences, attention mechanisms have been incorporated to NLP research area. It has already been proved that attention mechanism can greatly contribute to neural network based machine translation [Bahdanau, Cho and Bengio (2014)] and sequence encoding [Yin, Schütze, Xiang et al. (2015); Yang, Yang, Dyer et al. (2016); Lin, Shen, Liu et al. (2016); Wang, Hamza and Florian (2017); Gong, Luo and Zhang (2017); Kim, Hong, Kang et al. (2018)] through better semantic context rememberance machanism.

Considering the excellent semantic context capturing of attention machanism, our work follows this research stream and proposes a hybrid approach named deep feature fusion model to further memorize semantic features. Our hybrid deep feature fusion model consists of multiple separate sequence encoding approaches and an aggregation component to integrate different encoding outcomes. Another motivation is to design an innovative loss function to prevent over-fitting issue which is a must in modelling. In the most existing deep learning applications, cross entropy is so commonly used as loss function to train models. This approach sourcing from maximum likelihood estimation is very likely to categorize as 0 or 1 even with input noise, which could cause over-fitting issue. However, to our best knowledge, there is very little work on designing new loss functions. This paper also contributes a new alternative loss function incorporating confidence interval and threshold setting.

The main contributions are summarized as follows:

- We proposed a novel sentence encoding method named deep feature fusion to better capture semantic context via the integration of multiple sequence encoding approaches.
- We integrated the deep feature fusion approach into the most popular deep learning architecture for sentence matching task. The new architecture mainly consists of embedding layer, deep feature fusion layer, matching layer and prediction layer.
- We proposed a new loss function considering confidence interval and threshold setting, which preserves the loss caused by fuzzy instances and focus more on indistinguishable instances.
- We evaluated our approach on the common Chinese semantic matching corpus LCQMC, and the public English semantic matching corpus Quora. The results demonstrated that our approach outperforms other public advanced models on LCQMC and won the second place on Quora corpus, which really proved the superiority of our proposed method.
- We also open sourced our models in GitHub[4] to benefit the whole NLP community, see footnote for your reference.

The rest of the paper is structured as follows. We introduce the related work about sentence semantic matching in Section 2, and propose our new deep feature fusion model and architecture in Section 3. Section 4 demonstrated the empirical experimental results, followed by the conclusion in Section 5.

2 Related work

Sentence pair modeling has received extensive attention in the last decade. Many complex natural language processing tasks can be simplified into sentence semantic matching tasks. For example, information retrieval is to match query terms to documents, QA system is to match the query sentence to the given questions within question-answer pairs.

Probably a decade ago, SSM research mainly focused on latent semantic analysis and basic syntactic similarity calculation [Das and Smith (2009); Surdeanu, Ciaramita and Zaragoza (2011); Bär, Biemann, Gurevych et al. (2012); Meng, Lu, Zhang et al. (2018); Lu, Wu, Jian et al. (2018)]. With the advent of more competitive deep learning techniques, a lot more attention also turned to deep learning based SSM [Zhang, Lu, Ou et al. (2019)]. For example, deep structured semantic models [Huang, He, Gao et al. (2013)] and siamese networks [Mueller and Thyagarajan (2016)] simply encoded two sentences via a fully connected CNN or RNN, then calculated sentence matching similarity without considering the local phrase structure existing in the sentence. Further that, Wan et al. proposed BiLSTM to encode the query sentence and the candidate sentence, then calculated if LSTM's hidden layer output matched or not. BiLSTM contributed significantly to SSM research area, because of its capable of handling

[4] https://github.com/XuZhangp/Work2019_DFF_SSM

temporal relationships between sentences, capturing long-term word dependencies, and examining the meaning of each word in different contexts [Wan, Lan, Guo et al. (2016)]. Pang et al. constructed three superposition matching matrices to consider the word-word relationship between sentences. Similar to image application, CNN is then applied to extract significant features from these matching matrices [Pang, Lan, Guo et al. (2016)]. To further retain long-term context, Bai et al. proposed a Temporal Convolutional Network (TCN) [Bai, Kolter and Koltun (2018)]. Experiments showed that TCN based sentence encoding cannot only memorize long-term context more realistically, but also outperforms LSTM [Hochreiter and Schmidhuber (1997)].

To further improve matching performance, interactive mechanism like attention emerges to mine the connections between different words in sentences using a more elaborate structure. For example, Wang et al. applied four different methods to consider the interaction between sentences [Wang, Hamza and Florian (2017)] with the idea of sequential sentences shouldn't only be considered as one direction. Tomar et al. modified the input representation of the decomposed attention model, using n-gram character embedding instead of word embedding. They then pre-trained all model parameters on Paralex [Fader, Zettlemoyer and Etzioni (2013)], a noisy auto-collected corpus of problem definitions, and fine-tuned the parameters on the Quora dataset [Tomar, Duque, Täckström et al. (2017)]. Gong et al. proposed a complex deep neural network [Gong, Luo and Zhang (2017)] to consider the interactions between different words in the same sentence and to retain the original features through DenseNet [Huang, Liu, Van Der Maaten et al. (2017)] model. Kim et al. proposed a closely connected common attention RNN, which preserves the original information from the lowest level to the highest level. In each block of the stacked RNN, the interaction between two sentences is achieved through common attention. Because stacked RNN rapidly increases the number of arguments, autocoder [Kim, Hong, Kang et al. (2018)] has also been used to compress them. Subramanian et al. tried to combine different sentence representations of learning objectives into a single multi-task framework [Subramanian, Trischler, Bengio et al. (2018)].

Although the above deep learning models have already achieved good performance on sentence encoding, there are still two challenges to be addressed. First, the common RNN or CNN methods still have the problem of capturing long-term context. Second, attention mechanism and multi-granularity matching strategy might lose features in the sentence encoding process. As for the above challenges, our proposed deep feature fusion model and extended deep learning architecture clearly demonstrated absolute advantages in sentence encoding.

3 Deep feature fusion model

This Section is mainly to detail our proposed deep feature fusion model, starting from introducing the model architecture, followed by sub-modules of the model architecture including embedding layer, deep feature fusion layer, matching layer, prediction layer and the improved loss function.

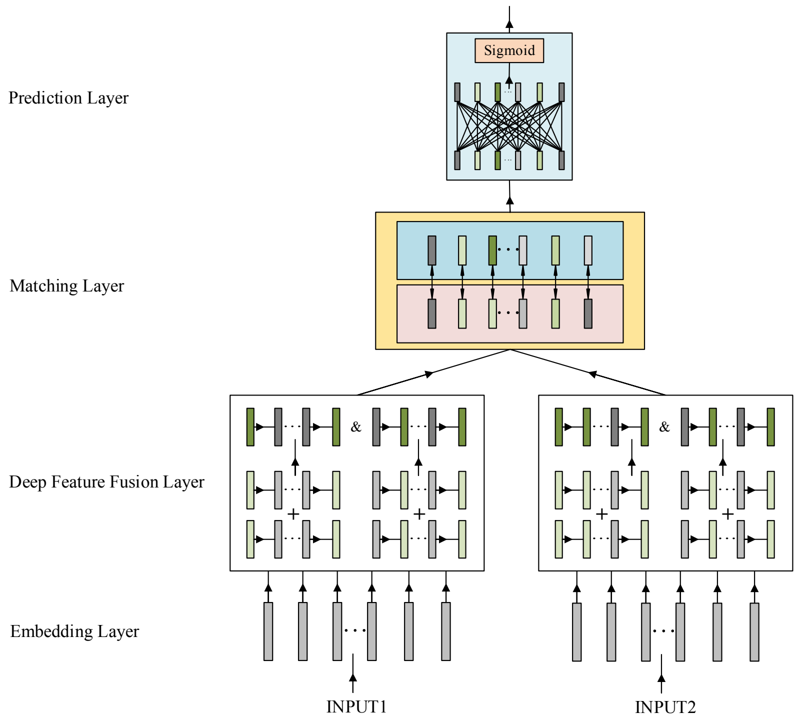

Figure 1: Model architecture of sentence matching

3.1 Model architecture

We first introduce the model architecture in Fig. 1. As seen, this model architecture includes multiple connected layers such as embedding layer, deep feature fusion layer, matching layer and prediction layer. Given input sentences, we first embed words and phrases through embedding layer, then transit the output of embedding layer to deep feature fusion layer to extract and hybrid semantic features. This feature fusion layer is our key contribution module to improve the semantic encoding performance. After semantic feature extraction, the encoding output of the deep feature fusion module will input to matching layer, followed by a decoding process in the prediction layer via sigmoid function.

3.2 Embedding layer

This embedding layer converts tokens such as words or phrases into embedding vectors first, then constructs a sentence matrix representation out of the word and phrase embedding vectors. Basically, multiple embedding approaches, for example pre-trained word embeddings corpus, can be applied to map tokens (words and phrase) to embedded vectors. For the experiments on LCQMC, we use word and character embedding approach to embed tokens by randomly initializing the word (character) vector. For Quora, we apply pre-trained 300 dimensions word embedding vectors from Glove [Pennington, Socher and Manning (2014)] to map sentence tokens to high-dimensional embedding vectors.

3.3 Deep feature fusion layer

This architecture includes a key encoding process named deep feature fusion as shown in Fig. 2. This deep feature fusion process starts from injecting the embedding matrix out of the embedding layer. Actually, the output of embedding layer for SSM task consists of two token embedding vectors, i.e., $P=[\mathrm{p}_1,\cdots,\mathrm{p}_M]$ and $Q=[q_1,\cdots,q_M]$, representing the given two sentences to match. We then apply LSTM and Dense wrapped in Time Distribtued to encode the embedding matrix. To reach the best performance, we separately apply LSTM and Dense twice on a sentence embedding matrix. The encoding will then result in four different outputs in two categories LSTMs and Denses.

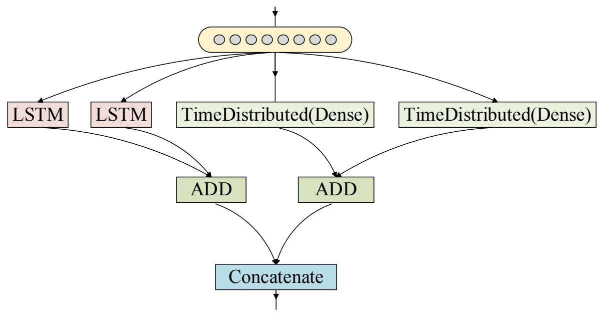

Figure 2: Deep feature fusion

For a given sentence P, two LSTMs outputs can be calculated using Eq. (1) and Eq. (2), similarly to calculate Denses outputs through Eq. (3) and Eq. (4).

$$\overrightarrow{h_i^p}=\overrightarrow{LSTM}(\overrightarrow{h_{i-1}^p},p_i) \tag{1}$$

$$\overrightarrow{h'^p_i}=\overrightarrow{LSTM}(\overrightarrow{h_{i-1}^p},p_i) \tag{2}$$

$$\overrightarrow{t_i^p}=\overrightarrow{TimeDistributed(Dense)}(\overrightarrow{t_{i-1}^p},p_i) \tag{3}$$

$$\overrightarrow{t'^p_i}=\overrightarrow{TimeDistributed(Dense)}(\overrightarrow{t_{i-1}^p},p_i) \tag{4}$$

The final step of the deep feature fusion model is to aggregate the encoding outputs together including two simple approaches: ADD and Concatenate. The ADDing approach will aggregate the results with same data structure together, Eq. (5) for LSTMs and Eq. (6) for Denses. After ADDing operation for same structure data, we then perform a concatenate operation, show in Eq. (7) to hybrid LSTM and Dense outcomes together.

$$\overrightarrow{H_i^p}=\overrightarrow{h_i^p}+\overrightarrow{h'^p_i} \tag{5}$$

$$\overrightarrow{T_i^p}=\overrightarrow{t_i^p}+\overrightarrow{t_i^p} \tag{6}$$

$$\overrightarrow{Connect_i^p}=[\overrightarrow{H_i^p},\overrightarrow{T_i^p}] \tag{7}$$

In this way, the embedding vector is further enhanced using ADDing operation within the same dimension data, and the memory semantic context is increasingly captured by concatenate operation for different dimension data.

Similar to sentence P, the same encoding methods can be applied for the matching sentence Q. The deep feature fusion outputs will then be forwarded to upper matching layer.

3.4 Matching layer

Taking the output of the deep feature fusion layer as input, matching layer applied three different matching strategies to calculate the similarity/dissimilarity between two sentence vectors P and Q as tensors. The first one is to calculate the absolute distance between the two sentence vectors as shown in Fig. (8)[5]. Second one is to multiply[6] vectors together using Eq. (9), the last strategy is to calculate the cosine[7] value for the two vector differences using Eq. (10). Finally, the three calculated tensors $C1_{ij}, C2_{ij}, C3_{ij}$ can be forwarded to the next prediction layer.

$$C1_{ij} = | A_{ij} - B_{ij} | \tag{8}$$

$$C2_{ij} = A_{ij} \times B_{ij} \tag{9}$$

$$C3 = \cos(A_{ij} - B_{ij}) \tag{10}$$

3.5 Prediction layer

The prediction layer itself is a deep neural network (DNN) classifier, including multiple sub-layers to fully extract sentence matching features out of the lower matching layer. In the DNN based prediction layer, encoding and matching modules map the input to the hidden feature space. The fully connected network maps the learned distributed feature representation to the sample tag space. This prediction layer actually consists of three dense sub-layers, with 600 dimensions for the first two denses and 1 dimension for the last dense. Each dense layer is followed by a dropout, relu and normalization modules. Then last sub-layer dense to be classified using sigmoid activation function.

3.6 Improved loss function

As a popular evaluation metric, mean square error (MSE) as in Eq. (11) measures the average of the squares of the errors, that is, the average squared difference between the estimated values y_{true} and what is estimated y_{pred}. MSE values closer to zero are better.

$$L_{mse} = \frac{1}{n} \sum_{i=1}^{n} (y_{true} - y_{pred})^2 \tag{11}$$

As we know that the gradient of MSE loss will be larger when the loss value is higher. It drops as the loss approaches to zero, making it more accurate at the end of training. But one disadvantage of using MSE is that its partial derivative value is very small when the output probability value is close to zero or close to 1, which may cause the partial

[5] https://github.com/keras-team/keras/blob/master/keras/backend/tensorflowbackend.py

[6] https://github.com/keras-team/keras/blob/master/keras/layers/merge.py

[7] https://github.com/keras-team/keras/blob/master/keras/backend/tensorflowbackend.py

derivative value to almost disappear when the model first starts training. This results in a very slow rate at the beginning of the model. This issue can be partly solved by fitting the $ones\_like$ [8] distribution as shown in Eq. (12).

$$L_{mse} = (1-k)\frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2 + k\frac{1}{n}\sum_{i=1}^{n}((ones\_like / r) - y_{pred})^2 \tag{12}$$

where： $r > 0; [0,1] \ni k$.

However, cross-entropy in Eq. (13) as an alternative loss function won't have this problem. Cross-entropy is actually a very popular loss function applied in machine learning research to measure the similarity between prediction values and the original values. In machine learning research, coupled with sigmoid function, cross-entropy is capable to solve the above MSE problem regarding the very slow learning rate when the gradient decreases, because the learning rate can be controlled by the output error.

$$L_{crossentropy} = -\sum_{i=1}^{n} y_{true} \log y_{pred} \tag{13}$$

To avoid the over-fitting problem for training, we update the cross entropy loss function in Eq. (14) by introducing a unit step function $\theta(\mathrm{x})$.

$$L_{new_{crossentropy}} = -\sum_{i=1}^{n} \lambda(y_{true}, y_{pred})(y_{true} \log y_{pred} + (1 - y_{true}) \log(1 - y_{pred})) \tag{14}$$

where:

$$\theta(\mathrm{x}) = \begin{cases} 1, x > 0 \\ \frac{1}{2}, x = 0 \\ 0, x < 0 \end{cases} \tag{15}$$

and

$$\lambda(y_{true}, y_{pred}) = 1 - \theta(y_{true} - m)\theta(y_{pred} - m) - \theta(1 - y_{true} - m)\theta(1 - y_{pred} - m) \tag{16}$$

We can see that $L_{new_{crossentropy}}$ updated the correction term $\lambda(y_{true}, y_{pred})$ to the common used cross entropy. For a positive sample, that means $y_{true} = 1$, obviously we will have: $\lambda(1, y_{pred}) = 1 - \theta(y_{true} - m)$. In this situation, if $y_{pred} > m$, then $\lambda(1, y_{pred}) = 0$, then the cross entropy is automatically 0 (to reach the minimum value). Otherwise if $y_{pred} < m$, there will be $\lambda(1, y_{pred}) = 1$. So, for our proposed new cross entropy, if the positive sample is bigger than m, it won't be updated because of reaching the minimum value. Otherwise if the sample value is smaller than m, it will be updated. Similarly, for a negative sample, if the output of the negative sample is smaller than $1 - m$, then won't be

[8] https://github.com/keras-team/keras/blob/master/keras/backend/tensorflowbackend.py

updated, otherwise if bigger than $1-m$, will continue to update. To improve classification performance and speed up the training speed, we hybrid the MSE, modified MSE and our proposed cross entropy together as our final loss function shown in Eq. (17).

$$L_{ours} = (1-n)L_{mse} + pL_{new_{mse}} + qL_{new_{crossentropy}} \tag{17}$$

where: $n = p+q; 1-n > 0; [0,1] \ni p,q,n$.

The following experiments on public data sets also demonstrated that our proposed loss function outperforms others.

4 Experiments and results

This Section starts with public data sets introduction, and our experiments settings to evaluate our proposed model, followed by results comparisons.

4.1 Data sets

We respectively evaluated our model on two public data sets, LCQMC and Quora. LCQMC is a large-scale Chinese question matching corpus released to the public by Liu et al. [Liu, Chen, Deng et al. (2018)], and Quora is one of the largest English question pair corpus released to the public by Chen et al. [Chen, Zhang, Zhang et al. (2018)].

For LCQMC data set, we pre-process the sentences using Chinese word segmentation, because Chinese doesn't automatically have spaces like English. For the experiments on LCQMC data set, we apply Word2vec technique [Mikolov, Chen, Corrado et al. (2013)] to train word vectors. For the experiments on data set Quora, we maintain a consistent approach similar to Wang et al. [Wang, Hamza and Florian (2017)], using pre-trained word vectors (300D Glove 840B) [Pennington, Socher and Manning (2014)].

4.1.1 LCQMC

LCQMC is a generic corpus mainly for intent matching, which contains a training set of 238,766 question pairs, a development set of 8,802 question pairs, and a test set of 12,500 question pairs. This data set mainly consists of two parts, a sentence pair including two sentences, and a binary matching label indicating if the two sentences matched or not. To better illustrate this data set, we randomly select a few examples as shown in Tab. 1. From these example sentence pairs, we can clearly see that the two matched sentences should be similar semantically.

Table 1: LCQMC corpus examples

Sentence Pairs	Semantic Match?
S1: 这个表情叫什么	
EN: What is this expression	NO
S2: 这个猫的表情叫什么	
EN: What is the cat's expression	
S3: 一只蜜蜂落在日历上（打一成语）	
EN: A bee lands on the calendar (Guess an idiom)	YES

S4: 一只蜜蜂停在日历上（猜一成语） EN: A bee sits on a calendar (Guess an idiom)	
S5: 一盒香烟不拆开 能存放多久？ EN: How long can be stored that a box of cigarettes without being opened? S6: 一条没拆封的香烟能存放多久。 EN: How long can be stored that a cigarette unopened.	YES
S7: 什么是智能手环 EN: What is a smart bracelet? S8: 智能手环有什么用 EN: What is the use of smart bracelet?	NO

4.1.2 Quora

The Quora corpus contains over 400,000 question pairs, and each question pair is annotated with a binary label indicating whether the two questions are paraphrase of each other. To better understand the data set, we also randomly choose some examples shown in Tab. 2

Table 2: Quora corpus examples

Sentence Pairs	Semantic Match?
S9: How do I get funding for my web based startup idea? S10: How do I get seed funding pre product?	YES
S11: Is honey a viable alternative to sugar for diabetics? S12: How would you compare the United States euthanasia laws to Denmark?	NO
S13: How can I stop my depression? S14: What can I do to stop being depressed?	YES
S15: Why is Mia Khalifa suddenly so popular? S16: What are some tricks male porn stars have to last longer?	NO

4.2 Experiments setting

To reproduce the experimental outcomes, here is how we set up the experiments. In the embedding layer, the embedding dimension is set to 300. In the deep feature fusion module we set dimension to 300. In the prediction layer, the widths of the three dense layers are 600, 600 and 1. Dropout in deep feature fusion layer is 0.1 and in prediction layer is 0.5. Both of the deep feature fusion and prediction layer use relu as the activation function, except the last dense applies sigmoid function for classification. We also tested different parameter combinations for loss function, here is the optimal parameter values for the two data sets. For Quora, p, q are both set to 0.15 and m to 0.6. But for LCQMC, we set p, q both to 0.35 and m to 0.7.

4.3 Proposed model variations

- DFF^{m} is the baseline model. This model uses deep feature fusion to extract sentence eigenvalues, as in Section 3.3, and the interactive matching model, as Section 3.4. This model applies the MSE loss function which is shown in Eq. (11).
- DFF^{im} is similar with DFF^{m}，but with the improved MSE loss function as Eq. (12).
- DFF^{c} is similar with DFF^{m} as well，but using cross-entropy loss function as Eq. (13).
- DFF^{ic} is similar with DFF^{m}，but using the improved cross-entropy loss function as Eq. (14).
- DFF^{o} is also similar too，but with the hybrid version of loss function including MSE and cross-entropy as Eq. (17).

Since LCQMC data set is a Chinese corpus, we can encode sentences from the perspective of characters and words. Therefore, each of the above five models can be further customized to two sub-models. For example, DFF^{m} becomes DFF^{m} char and DFF^{m} word. But Quora data set is encoded only from word level.

4.4 Experiments on LCQMC

A comparison of our work with some of the existing work such as WMD, CBOW, CNN, BiLSTM, BiMPM Liu et al. [Liu, Chen, Deng et al. (2018)] is shown in Tab. 3. Our model DFF^{o} obviously outperforms the existing models WMD, CBOW, CNN, BiLSTM, BiMPM both on character and word levels.

Table 3: Experimental results on LCQMC

Methods	Precision	Recall	F_1-score	Accuracy
Baseline $(WMD)_{char}$	67.0	81.2	73.4	70.6
baseline $(WMD)_{word}$	64.4	78.6	70.8	60.0
C_{wo}	61.1	83.6	70.6	70.7
C_{ngram}	52.3	89.3	66.0	61.2
D_{edt}	46.5	86.4	60.5	52.3
S_{cos}	60.1	88.7	71.6	70.3
$CBOW_{char}$	66.5	82.8	73.8	70.6
$CBOW_{word}$	67.9	89.9	77.4	73.7
CNN_{char}	67.1	85.6	75.2	71.8
CNN_{word}	68.4	84.6	75.7	72.8
$BiLSTM_{char}$	67.4	91.0	77.5	73.5
$BiLSTM_{word}$	70.6	89.3	78.92	76.1
$BiMPM_{char}$	77.6	93.9	85.0	83.4
$BiMPM_{word}$	<u>77.7</u>	93.5	84.9	83.3
Our Models	Precision	Recall	F_1-score	Accuracy
DFF^{m}_{char}	76.64	95.13	84.84	83.06
DFF^{m}_{word}	75.83	94.56	84.13	82.21
DFF^{im}_{char}	77.32	94.49	85.01	83.40
DFF^{im}_{word}	76.51	94.83	84.65	82.86

DFF^{c}_{char}	76.27	95.29	84.67	82.81
DFF^{c}_{word}	75.89	**95.32**	84.46	82.51
DFF^{ic}_{char}	77.23	94.72	85.05	83.41
DFF^{ic}_{word}	77.02	94.63	84.88	83.19
DFF^{o}_{char}	**78.58**	93.88	**85.51**	**84.15**
DFF^{o}_{word}	77.69	94.08	85.06	83.53

4.4.1 Comparison with exiting models

From the character level, compared with WMD, CBOW, CNN, BiLSTM and BiMPM, our best model DFF^{o} improves the precision metric by 11.58%, 12.08%, 11.48%, 11.18%, 0.98%, recall by 12.68%, 11.08%, 8.28%, 2.88%, -0.02%, F_1-score by 12.11%, 11.71%, 10.31%, 8.01%, 0.51%, and accuracy by 13.55%, 13.55%, 12.35%, 10.65%, 0.75%.

We also respectively compare with models WMD, C_{wo}, C_{ngram}, D_{edt}, S_{cos}, CBOW, CNN, BiLSTM and BiMPM from word level. The comparison result shows that the precision is improved by 13.29%, 16.59%, 25.39%, 31.19%, 17.59%, 9.79%, 9.29%, 7.09%, -0.01%, recall is improved by 17.28%, 10.48%, 4.78%, 7.68%, 5.38%, 4.18%, 9.48%, 4.78%, 0.58%, F_1-score is improved by 14.26%, 14.46%, 19.06%, 24.56%, 13.46%, 7.66%, 9.36%, 6.14%, 0.16%, and accuracy is improved by 23.53%, 12.83%, 22.33%, 31.23%, 13.23%, 9.83%, 10.73%, 7.43%, 0.23%.

From Tab. 3, we can clearly see that our model works best at both word and character level on precision, recall, F_1-score and accuracy metrics.

4.4.2 Comparison with model variations

In this subsection, we compare our proposed model variations with different loss functions. The models also have word and character levels. First from the character level, compared to DFF^{m}_{char}, DFF^{im}_{char}, DFF^{c}_{char} and DFF^{ic}_{char}, the model DFF^{o}_{char} improves precision by 1.64%, 1.26%, 2.31%, 1.35%, F_1-score by 0.67%, 0.5%, 0.84%, 0.46%, accuracy by 1.09%, 0.75%, 1.34%, 0.74%. From the word level, compared to DFF^{m}_{word}, DFF^{im}_{word}, DFF^{c}_{word} and DFF^{ic}_{word}, and the model DFF^{o}_{word} improves precision by 1.86%, 1.18%, 1.85%, 0.67%, F_1-score by 0.93%, 0.41%, 0.6%, 0.18%, accuracy by 1.32%, 0.67%, 1.02%, 0.34%. The comparison results clearly show that the improved hybrid loss function has achieved the best performance. In addition, the outcome also shows character encoding is better than word encoding.

4.5 Experiments on Quora

We also evaluate our models on the Quora data set with result shown in Tab. 4. We compare it with the most advanced models available today, as Tab. 2. CNN, LSTM, L.D.C, BiMPM from wang et al. [Wang, Hamza and Florian (2017)], and FFNN, DECATT from Tomar et al. [Tomar, Duque, Täckström et al. (2017)], and DIIN is the work of Gong et al. [Gong, Luo and Zhang (2017)]. Experimental results show that our proposed models are still competitive on the Quora data set.

Table 4: Experimental results on Quora

Methods	$\text{Dev}_{\text{accuracy}}$	$\text{Test}_{\text{accuracy}}$
Siamese-CNN	-	79.60
Multi-perspective CNN	-	81.38
Siamese-LSTM	-	82.58
Multi-perspective LSTM	-	83.21
L.D.C	-	85.55
BiMPM	88.69	88.17
$\text{FFNN}_{\text{word}}$	85.07	84.35
$\text{FFNN}_{\text{char}}$	86.01	85.06
$\text{DECATT}_{\text{word}}$	86.04	85.27
$\text{DECATT}_{\text{glove}}$	87.42	86.52
$\text{DECATT}_{\text{char}}$	87.78	86.84
$\text{DECATT}_{\text{paralex-char}}$	87.80	87.77
pt-$\text{DECATT}_{\text{word}}$	88.44	87.57
pt-$\text{DECATT}_{\text{char}}$	<u>88.89</u>	88.40
DIIN	**89.44**	**89.06**
Our Models	$\text{Dev}_{\text{accuracy}}$	$\text{Test}_{\text{accuracy}}$
DFF^{m}	88.21	88.07
DFF^{im}	88.53	87.97
DFF^{c}	88.51	88.51
DFF^{ic}	87.77	87.54
DFF^{o}	88.61	<u>88.83</u>

5 Conclusion

We proposed a new deep neural network based sentence matching model with great performance achievement on two public data sets LCQMC and Quora. In this model, we proposed an innovative sentence encoding structure named deep feature fusion to better capture sentence's eigenvalues. At the mean time, we also proposed a hybrid loss function to better determine confidence interval and threshold setting for classification performance improvement. The experiments demonstrated that our proposed model outperforms the most advanced available models so far, which is contributed from the proposed deep feature fusion module. Additionally, we compared our model variations with different loss functions. The comparison outcome showed that the proposed hybrid loss function integrating MSE and cross entropy performs well on the two public data sets.

Acknowledgement: The research work is supported by National Nature Science Foundation of China under Grant No. 61502259, National Key R&D Program of China under Grant No. 2018YFC0831704 and Natural Science Foundation of Shandong Province under Grant No. ZR2017MF056.

References

Bahdanau, D.; Cho, K.; Bengio, Y. (2014): Neural machine translation by jointly learning to align and translate. arXiv: 1409.0473.

Bai, S.; Kolter, J. Z.; Koltun, V. (2018): An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv: 1803.01271.

Bär, D.; Biemann, C.; Gurevych, I.; Zesch, T. (2012): Ukp: computing semantic textual similarity by combining multiple content similarity measures. *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, vol. 1; *Proceedings of the Main Conference and the Shared Task*, vol. 2; *Proceedings of the Sixth International Workshop on Semantic Evaluation*, vol. (1, 2), pp. 435-440.

Blunsom, P.; Grefenstette, E.; Kalchbrenner, N. (2014): A convolutional neural network for modelling sentences. arXiv:1404.2188.

Chen, Z.; Zhang, H.; Zhang, X.; Zhao, L. (2018): Quora question pairs. https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F. et al. (2014): Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv:1406.1078.

Conneau, A.; Schwenk, H.; Barrault, L.; Lecun, Y. (2016): Very deep convolutional networks for text classification. *arXiv:1606.01781.*

Das, D.; Smith, N. A. (2009): Paraphrase identification as probabilistic quasi-synchronous recognition. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, vol. 1, no. 1, pp. 468-476.

Fader, A.; Zettlemoyer, L.; Etzioni, O. (2013): Paraphrase-driven learning for open question answering. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 1608-1618.

Gers, F. A.; Schmidhuber, J. (2000): Recurrent nets that time and count. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 3, pp. 189-194.

Gong, Y.; Luo, H.; Zhang, J. (2017): Natural language inference over interaction space. arXiv:1709.04348.

Greff, K.; Srivastava, R. K.; Koutník, J.; Steunebrink, B. R.; Schmidhuber, J. (2017): LSTM: a search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222-2232.

Hochreiter, S.; Schmidhuber, J. (1997): Long short-term memory. *Neural computation*, vol. 9, no. 8, pp. 1735-1780.

Hu, B.; Lu, Z.; Li, H.; Chen, Q. (2014): Convolutional neural network architectures for matching natural language sentences. *Advances in Neural Information Processing Systems*, pp. 2042-2050.

Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K. Q. (2017): Densely connected convolutional networks. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2261-2269.

Huang, P. S.; He, X.; Gao, J.; Deng, L.; Acero, A. et al. (2013): Learning deep structured semantic models for web search using clickthrough data. *Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management*, pp. 2333-2338.

Jozefowicz, R.; Zaremba, W.; Sutskever, I. (2015): An empirical exploration of recurrent network architectures. *Proceedings of the International Conference on Machine Learning*, pp. 2342-2350.

Kim, S.; Hong, J. H.; Kang, I.; Kwak, N. (2018). Semantic Sentence Matching with Densely-connected Recurrent and Co-attentive Information. arXiv: 1805.11360.

Kim, Y. (2014): Convolutional neural networks for sentence classification. arXiv:1408.5882.

Lin, Y.; Shen, S.; Liu, Z.; Luan, H.; Sun, M. (2016): Neural relation extraction with selective attention over instances. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 2124-2133.

Liu, X.; Chen, Q.; Deng, C.; Zeng, H.; Chen, J. et al. (2018). LCQMC: a large-scale chinese question matching corpus. *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1952-1962.

Liu, Y.; Sun, C.; Lin, L.; Wang, X. (2016): Learning natural language inference using bidirectional LSTM model and inner-attention. arXiv:1605.09090.

Lu, W.; Wu, H.; Jian, P.; Huang, Y.; Huang, H. (2018): An empirical study of classifier combination based word sense disambiguation. *IEICE Transactions on Information and Systems*, vol. 101, no. 1, pp. 225-233.

Meng, F.; Lu, W.; Zhang, X.; Jin, Y. (2018): Word sense disambiguation method based on hownet and graph model. *Journal of Qilu University of Technology*, vol. 32, no. 6, pp. 66-73.

Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. (2013): Efficient estimation of word representations in vector space. arXiv: 1301.3781.

Mou, L.; Peng, H.; Li, G.; Xu, Y.; Zhang, L. et al. (2015): Discriminative neural sentence modeling by tree-based convolution. arXiv:1504.01106.

Mueller, J.; Thyagarajan, A. (2016): Siamese Recurrent Architectures for Learning Sentence Similarity. *Proceedins of the thirtieth AAAI Conference on Artificial Intelligence*, vol. 16, pp. 2786-2792.

Pang, L.; Lan, Y.; Guo, J.; Xu, J.; Wan, S. et al. (2016): Text matching as image recognition. *Proceedins of the thirtieth AAAI Conference on Artificial Intelligence*, pp. 2793-2799.

Pennington, J.; Socher, R.; Manning, C. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1532-1543.

Qiu, X.; Huang, X. (2015): Convolutional neural tensor network architecture for community-based question answering. *Proceedings of International Joint Conferences on Artificial Intelligence*, pp. 1305-1311.

Subramanian, S.; Trischler, A.; Bengio, Y.; Pal, C. J. (2018): Learning general purpose distributed sentence representations via large scale multi-task learning. arXiv:

1804.00079.

Surdeanu, M.; Ciaramita, M.; Zaragoza, H. (2011): Learning to rank answers to nonfactoid questions from web collections. *Computational Linguistics*, vol. 37, no. 2, pp. 351-383.

Tan, M.; Santos, C. D.; Xiang, B.; Zhou, B. (2015): LSTM-based deep learning models for non-factoid answer selection. arXiv: 1511.04108.

Tomar, G. S.; Duque, T.; Täckström, O.; Uszkoreit, J.; Das, D. (2017): Neural paraphrase identification of questions with noisy pretraining. arXiv: 1704.04565.

Wan, S.; Lan, Y.; Guo, J.; Xu, J.; Pang, L. et al. (2016): A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, vol. 16, pp. 2835-2841.

Wang, Z.; Hamza, W.; Florian, R. (2017): Bilateral multi-perspective matching for natural language sentences. arXiv:1702.03814.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A. et al. (2016): Hierarchical attention networks for document classification. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480-1489.

Yin, W.; Schütze, H. (2015): Multigrancnn: an architecture for general matching of text chunks on multiple levels of granularity. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, vol. 1, pp. 63-73.

Yin, W.; Schütze, H.; Xiang, B.; Zhou, B. (2016). ABCNN: attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 259-272.

Zhang, S.; Zhang, X.; Wang, H.; Cheng, J.; Li, P. et al. (2017): Chinese medical question answer matching using end-to-end character-level multi-scale CNNs. *Applied Sciences*, vol. 7, no. 8, pp. 767.

Zhang, Y.; Lu, W.; Ou, W.; Zhang, G.; Zhang, X. et al. (2019): Chinese medical question answer selection via hybrid models based on CNN and GRU. *Multimedia Tools and Applications.*