Related Questions Detection Model in Stack Overflow based on Semantic Matching

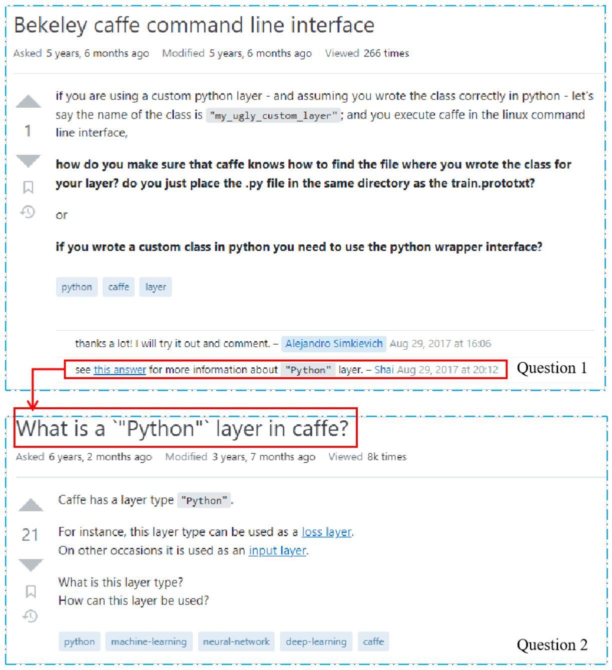
Shizhao Huang[†],Yimin Wu[‡§], Jinwei Lu[‡], Chao Deng[§]
[†]School of Software Engineering, South China University of Technology, Guangzhou, China
[‡]School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[§]School of Computer Science and Engineering, Guangdong Ocean University at yangjiang, Yangjiang, China
se_hsz@mail.scut.edu.cn, csymwu@scut.edu.cn,
cskilljl@mail.scut.edu.cn, dengchao@gdou.edu.cn

***Abstract*—Stack Overflow is a widely-used community Q&A website for programming-related queries. In such a platform, providing related questions as suggestions to the users can significantly enhance their search experience. Although there are many approaches based on deep learning that can automatically predict the relatedness between questions, those approaches are limited because the semantic and interaction features of the sentences may be lost. In this paper, we propose a novel method to predict the relatedness between questions based on semantic matching. We adopt the Interaction Feature Extractor to capture the interaction information and fuse it through a fusion mechanism to enhance the interaction between questions. Our experimental results demonstrate that our proposed method achieves state-of-the-art performance in terms of Precision, Recall, and F1-score evaluation metrics, outperforming the baseline approaches. Furthermore, we show that our model also performs well in other semantic matching tasks in software fields, indicating its generalization ability and robustness.**

***Index Terms*—Stack Overflow, Question Relatedness, Deep Learning, Semantic Matching**

I. Introduction

Stack Overflow is a Programming Community-based Question Answering (PCQA) forum that functions as a platform for developers to seek solutions to programming issues and exchange knowledge in their respective fields. Over time, Stack Overflow has accumulated a large number of questions, with many duplicate and related questions. Figure 1 illustrates an example of a pair of related questions. They are discussing different but related issues, which means that the answers in question 2 contribute to the solution of question 1. It is worth mentioning that typically, developers manually share related questions through URL links, which are identified after the question has been posted. If developers can find related questions when seeking solutions to new issues, they can leverage the existing answers to efficiently resolve their problems. Meanwhile, these available answers enable developers to avoid posting duplicate issues, thereby facilitating the maintenance of the website. The website contains a vast array of questions, and given the possibility of expressing the same question in multiple ways, manual identification methods prove to be inefficient and time-consuming. Hence,



Bekeley caffe command line interface

Asked 5 years, 6 months ago Modified 5 years, 6 months ago Viewed 266 times

1

if you are using a custom python layer - and assuming you wrote the class correctly in python - let's say the name of the class is `"my_ugly_custom_layer"`; and you execute caffe in the linux command line interface,

how do you make sure that caffe knows how to find the file where you wrote the class for your layer? do you just place the .py file in the same directory as the train.prototxt?

or

if you wrote a custom class in python you need to use the python wrapper interface?

python caffe layer

thanks a lot! I will try it out and comment. – Alejandro Simkievich Aug 29, 2017 at 16:06

see this answer for more information about `"Python"` layer. – Shai Aug 29, 2017 at 20:12

Question 1

What is a `"Python"` layer in caffe?

Asked 6 years, 2 months ago Modified 3 years, 7 months ago Viewed 8k times

21

Caffe has a layer type `"Python"`.

For instance, this layer type can be used as a loss layer.
On other occasions it is used as an input layer.

What is this layer type?
How can this layer be used?

python machine-learning neural-network deep-learning caffe

Question 2

Fig. 1 A sample of related question pair

it is necessary to propose an automated method to identify duplicate and related questions.

Due to its strong nonlinear fitting ability, deep learning can effectively extract semantic information, and in recent work, some researchers have employed deep learning to predict question relatedness[1,3,6,9]. Taking Pei *et al.*'s [3] work as an example, their method adopts BiLSTM to extract contextual information and employs the soft attention mechanism to extract the interaction features of words. However, their model only learns the interaction features of words and ignores the extraction of semantic and interaction features of sentences, which are crucial for predicting the semantic relevance between questions.

This paper conducts research based on the work of Pei *et al.* [3]. In order to leverage interaction features to predict the semantic relevance between questions, we propose a related questions Detection Model in Stack Overflow (named DMSO), which detects the related questions based on the semantic

DOI reference number: 10.18293/SEKE2023-163

relatedness between questions. To better extract the features of sentences, we introduce Interaction Feature Extractor to enhance the model's ability to extract interaction information and fuse it through a fusion mechanism to enhance the interaction between questions. To evaluate the proposed model, we conduct experiments on the public dataset built by Shirani *et al.* [6] and compare with the baseline models.

The main contributions of this paper are as follows:

- We proposed a novel semantic matching model to detect the semantic relatedness between questions, which is an automated method to identify duplicate and related questions.
- We implement an interaction feature extractor to capture the features of sentences to enhance the model's ability to extract interaction formation.
- We evaluate our approach on a public dataset, and the result shows that our model not only outperforms previous methods in the question relatedness prediction task but also has a good performance in duplicate question detection tasks in software engineering domains.

The rest of the paper is organized as follows. Section II briefly describes the related work of our study. Section III introduces the overall framework and technical details of our approach. Section IV describes the experimental settings and presents the experiment results. Section V concludes the paper and outlines future work.

II. Related work

A. Detecting the semantic relatedness between questions on programming community Q&A sites

Predicting the semantic relatedness between questions on programming community Q&A sites is beneficial to improve the efficiency of users in finding questions, and even helps them solve problems directly. Xu *et al.* [1] refer to a question and its answers in Stack Overflow as a knowledge unit, and divide these knowledge units into four classes: *duplicate*, *direct*, *indirect*, and *isolated*. In their paper, they proposed a CNN model, which adopts a convolutional neural network to extract contextual information in the knowledge units. They then calculate the semantic similarity of two contextual vectors by the cosine function and predict the class of knowledge unit pairs based on the semantic similarity. Fu *et al.* [8] proposed an SVM-based model(Tuning SVM), which uses word2vec [7] to obtain word embedding and adopts differential evolution (DE) as its tuning algorithm. In their study, tuning SVM with parameter tuning runs much faster than the CNN model. Following the study of Fu *et al.*, Xu *et al.* [9] proposed the Soft-Cos SVM model in their paper. The Soft-Cos SVM model calculates the soft cosine similarity based on Simbow to measure the distance between knowledge unit pairs and adopt SVM as the final classifier. In addition, Xu *et al.* built a dataset containing 40,000 pairs of knowledge units. Based on the work of Xu *et al.*, Shirani *et al.* [6] constructed a dataset with more than 300,000 pairs of knowledge units (hereafter, Knowledge Unit dataset). They also constructed two baseline models SOFTSVM and DOTBILSTM. SOFTSVM is similar to the previous SVM-based method, and it uses word2vec for word embedding of knowledge units, then calculates the cosine similarity, and finally predicts the semantic relatedness between knowledge units by SVM. DOTBILSTM uses BiLSTM to extract contextual information in the knowledge units, then calculates the inner product of the contextual vectors and obtains the probability distribution of each class using fully connected layers and softmax functions.

In recent work, Pei *et al.* [3] proposed the attention-based model ASIM. This model also adopts BiLSTM to encode local semantic information and obtain the interaction information between two knowledge units by soft attention mechanism. The experimental results show that ASIM is the state-of-the-art model in the dataset built by Shirani *et al.*. However, their approach was based on word-level interaction features between pairs of knowledge units through the soft attention mechanism. Therefore, these methods cannot sufficiently consider the semantic and interaction features at the sentence level, which is crucial for predicting the semantic relevance between questions. Therefore, we introduce a novel interaction method to improve the extraction of interaction features, hoping to achieve better performance in the task of predicting semantic relatedness between questions.

B. Duplicate question detection

Previous studies have shown [11] that the rapid growth in the number of duplicate questions is not conducive to website maintenance and it can lead to a decrease in the number of active users of community Q&A sites. AskUbuntu is another popular programming community Q&A site. Bogdanova *et al.*[14] construct a dataset with 30,000 question pairs from the AskUbuntu data dump and adopt CNN to detect duplicate questions. Based on the work of Bogdanova et al., Rodrigues *et al.*[15] released the AskUbuntu dataset's clean version. They proposed a hybrid deep convolutional network model DCNN. The deep learning approaches described above do not require manually designed features and have a higher accuracy rate compared to traditional automatic detection methods. However, these methods also fail to consider the interaction information of sentences. We will use the AskUbuntu dataset released by Rodrigues *et al.*[15] to explore the generalization performance of DMSO in similar tasks.

III. THE APPROACH

According to Xu *et al.* [1], a question in Stack Overflow and its answers is a knowledge unit (KU). Based on the degree of relatedness between two knowledge units from high to low, the relatedness types between them can be defined as the following four classes:

- *Duplicate*: The questions in the two knowledge units are duplicate questions.
- *Direct*: Information in one knowledge unit can directly solve the question in another knowledge unit.
- *Indirect*: The information in one knowledge unit is helpful to the solution of the question in another knowledge

unit, but the information alone cannot directly solve the question.

- *Isolated*: There is no semantic relatedness between the two knowledge units.

We treat this task as a multi-class classification problem. The input to the model is a pair of knowledge units, and the relatedness is predicted as one of the four classes mentioned above.

Figure 2 gives an illustration of the DMSO framework, which is mainly composed of the following five modules: (1) *Word Embedding Layer*, (2) *Word Encoding Layer*, (3) *Local Interaction Layer*, (4) *Global Interaction Fusion Layer*, and (5) *Prediction Layer*.

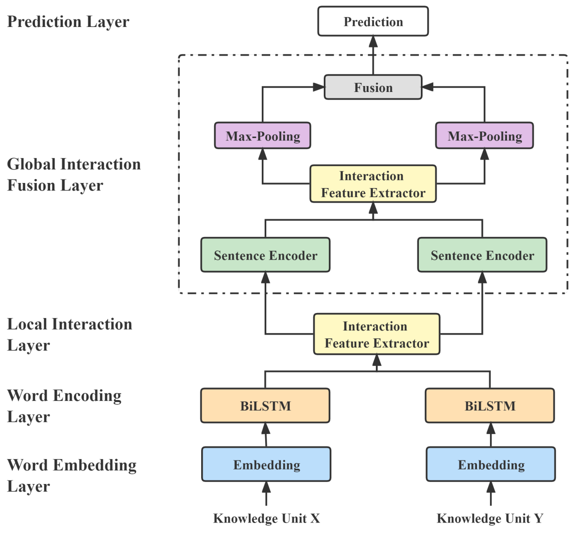


Fig. 2 The framework of DMSO

A. Input of the model

We concatenate the title, body, and answers to the question to form a textual sequence as the input text. And we apply some data pre-processing steps on the input text, including dividing by sentence, normalizing URLs and numbers, removing punctuation marks and stop words, splitting camel case words, stemming, and changing all words to lowercase. We use the pre-processed text sequences as the final input to the model. Suppose the text sequence of a knowledge unit is $KU = \{w_1, w_2, ..., w_n\}$, where n is the sequence length. The inputs of the model DMSO are text sequences KU_X and KU_Y, and the target of the model predicts the relatedness of KU_X and KU_Y. Since the two text sequences are treated symmetrically before the Prediction Layer, we will only present the processing of KU_X in the model to avoid repetition.

B. Word Embedding Layer

In this layer, we adopt word embedding techniques to convert words into their corresponding vector representations. The word embeddings pre-trained in the universal domain containing large amounts of data not related to software engineering, may lead to ambiguous representations of words [16]. Therefore, we use the title, body, and answers in Stack Overflow as the corpus, constructing a 300-dimensional word vector through word2vec [7]. Each word w_i in the knowledge unit is transformed into a vector representation x_i with 300 dimensions. Therefore, the knowledge units containing n words can be converted into corresponding matrix representation, which is the input of the word encoder of the model:

$$KU_X = x_1 \oplus x_2 \oplus ... \oplus x_n \tag{1}$$

Where $\oplus$ is the concatenation operator.

C. Word Encoding Layer

In this layer, we adopt BiLSTM to fuse local features into each word's original representation. BiLSTM is composed of a forward and a backward LSTM[4]. The semantic understanding of natural language words depends on the context. This means that the meaning of the word depends not only on what has been read before but also on what will be read. We adopt BiLSTM as the word encoder, which can capture local contextual dependencies from both forward and backward directions. Let $\overrightarrow{h}_i, i \in [1, n]$ denote the hidden state of the forward LSTM in the time step i, and the encoding process of BiLSTM is as follows:

$$\overrightarrow{h}_i = LSTM(\overrightarrow{h}_{i-1}, x_i), \forall i \in \{1, 2, ..., n\} \tag{2}$$

$$\overleftarrow{h}_i = LSTM(\overleftarrow{h}_{i+1}, x_i), \forall i \in \{1, 2, ..., n\} \tag{3}$$

$$X_i = [\overrightarrow{h}_i; \overleftarrow{h}_i], \forall i \in \{1, 2, ..., n\} \tag{4}$$

D. Local Interaction Layer

Most of the previous deep learning models used in the semantic matching task on PCQA sites were based on representation models. These approaches do not consider the interaction information between these two questions. In the field of neural language processing, the attention mechanism was first applied in a neural machine translation model [19]. By applying the attention mechanism to the questions relatedness prediction task, the model can effectively extract the interaction information between questions through inter-sentence alignment and obtain better performance. Furthermore, Tay *et al.* [2] have shown that by equipping the attention mechanism with a more flexible structure, the model can generate more robust representations.

Therefore, in this Layer, we introduce a novel interaction method to improve the extraction of interaction features through constructing interaction attention and difference attention. Figure 3 shows the workflow of how the Interaction Feature Extractor extracts the interaction information between two features.

We first conduct a dot product between X_i and Y_j, which are the output of two words in the knowledge unit KU_X and KU_Y from the Word Encoding Layer. After the weighted summation of the attention scores of all words in KU_X and KU_Y, we can obtain the interaction attention matrix A. And the difference attention D adopts a subtraction-based attention mechanism,

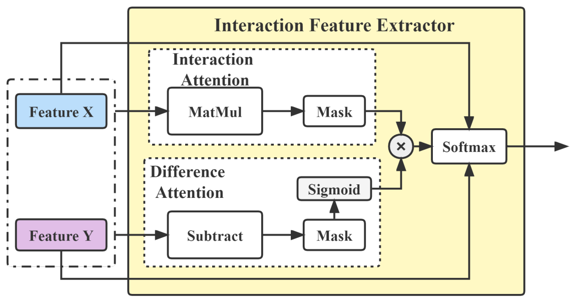


Fig. 3 The architecture of the Interaction Feature Extractor

which allows the model to pay attention to dissimilar parts between knowledge unit pairs by element-wise subtraction. The difference attention is introduced to better capture fine-grained interaction information between knowledge units.

$$A = X^T \cdot Y \tag{5}$$
$$D = \parallel X - Y \parallel \tag{6}$$

Where $\cdot$ denotes the inner production operation, $\parallel X - Y \parallel \in R^{n_x \times n_y}$, and n_x is the KU_X sequence length. We mask the two attention matrices and perform a $sigmoid$ operation on the difference attention D. Then we multiply A and D to get the attention matrix of the local interaction feature.

$$S^X = softmax(A \odot sigmoid(D)) \cdot Y \tag{7}$$
$$S^Y = softmax(A \odot sigmoid(D)) \cdot X \tag{8}$$

Where $softmax$ is a normalization function, $sigmoid$ is an activation function, and $\odot$ denotes element-wise multiplication.

E. Global Interaction Fusion Layer

Previous methods only consider local interaction features and ignore the extraction of global features, so it is difficult to capture the semantic information and dependencies at the sentence level. To address the problem that local interaction features are insensitive to the understanding of global information, we extract the semantic features and interaction features of sentences to represent the global interaction features in knowledge unit pairs through the Global Interaction Fusion Layer. We refer to the method of converting word vectors to sentence vectors in the paper [5], which proceeds as follows.

$$G^X = softmax(W_\alpha tanh(W_\beta S^X + b))S^X \tag{9}$$

Then we utilize the Interaction Feature Extractor to extract global interaction features, which can help capture the global relevance information between knowledge units. The global information complements the semantic relatedness that cannot be captured by local interaction.

$$F^X, F^Y = IFE(G^X, G^Y) \tag{10}$$

Where IFE denotes the operation of the Interaction Feature Extractor, which is the same as Equations (5-8) in the Local Interaction Layer. We convert the aggregation features F^X and F^Y into fixed-size vectors.

$$V^X_{pool} = Maxpooling(F^X) \tag{11}$$
$$V^Y_{pool} = Maxpooling(F^Y) \tag{12}$$

Then we concatenate the vectors in the fusion layer.

$$T = FFN[V^X_{pool}; V^Y_{pool}; V^X_{pool} - V^Y_{pool}; V^X_{pool} \odot V^Y_{pool}] \tag{13}$$

Where FFN denotes a single-layer feedforward network.

F. Prediction Layer

In the Prediction Layer, for the vectors T obtained from the Global Interaction Fusion Layer, we adopt a single-layer feedforward network to get the feature vector. Finally, we calculate the probability distribution of four classes with the softmax function.

IV. EXPERIMENT

In this section, we conducted some experiments to answer the following research questions.

A. Research Questions

RQ1: How effective is DMSO in predicting knowledge unit pairs of different classes?

DMSO adopts the semantic interaction-based approach, introducing sentence-level interaction features to complement the semantic relatedness that cannot be captured by local interaction, which is a significant departure from previous work. To investigate the effectiveness of our approach, we compare the performance of DMSO with the baseline from Pei et al.'s work [3]. To present results more accurately, we keep the results of all models to three decimal places.

RQ2: How much influence do the modules we proposed contribution to the improvement of DMSO?

Two important modules we proposed, including Global Interaction Fusion Layer and Interaction Feature Extractor, can help DMSO to capture the rich features of knowledge units by extracting both the semantic and interaction features of sentences. To evaluate their contributions, we perform the following ablation studies, consisting in (1) removing the Global Interaction Fusion Layer (GIF) and replacing it with the max pooling operation; (2) removing the Interaction Attention (IA) in two Interaction Feature Extractors; (3) removing the Difference Attention (DA) in two Interaction Feature Extractors; , and (4) removing two Interaction Feature Extractors (IFE). Then, we compare the revised model with the original model on the F1-score.

RQ3: Does DMSO work well in other semantic relevance tasks from software engineering domains?

Duplicate question detection in programming community Q&A sites is also a task to study semantic relevance in software engineering. In contrast to predicting question relevance, this task is a two-class classification problem where the evaluation metric is accuracy. We want to explore the generalization performance of DMSO through a different software engineering task.

B. Dataset

We carry out experiments on the dataset built by Shirani *et al.* [6]. This dataset focuses on Java-related knowledge units on Stack Overflow, and it contains 160,161 distinct knowledge units and 347,372 pairs of knowledge units with four types of relationships. In this paper, it is divided into three parts with the same proportion as in [6], that is, the training set containing 208, 424 knowledge unit pairs, the validation set containing 34, 736 knowledge unit pairs, and the test set containing 104, 212 knowledge unit pairs.

Moreover, to evaluate the generalization ability of DMSO in a similar task, we also run experiments on the duplicate question detection task in software engineering, using the clean version of the AskUbuntu dataset prepared by Rodrigues *et al.*[15]. In the AskUbuntu dataset, 24K question pairs are used for training, 6K for testing, and 1K for validation. The two classes in the AskUbuntu dataset are balanced, so there are an equal number of duplicate and non-duplicate question pairs. Since the dataset contains only questions and not answers, we concatenate the question titles and bodies as inputs to the model and modify the output of the prediction layer to 2 classes. The other parts of the model remain unchanged.

C. Evaluation Metric

We use the same evaluation metrics as in the previous work[3,6] to evaluate DMSO's performance. $Precision_i$ represents the proportion of knowledge unit pairs correctly classified as class i. $Recall_i$ is the percentage of all class i correctly classified. $F1\text{-}score_i$ is harmonic mean of $Precision_i$ and $Recall_i$.

$$Precision_i = \frac{TruePositive_i}{TruePositive_i + FalsePositive_i} \tag{14}$$

$$Recall_i = \frac{TruePositive_i}{TruePositive_i + FalseNegative_i} \tag{15}$$

$$F1 - score_i = \frac{2 \times Precision_i \times Recall_i}{Precision_i + Recall_i} \tag{16}$$

D. Implementation Details

The deep learning framework we use is PyTorch 1.9.1 with CUDA 10.2 as the GPU computing platform, and all experiments are implemented in a python 3.7 environment. For our model, 300-dimensional word embeddings are used according to experience. We set the hidden size as 200 for all feedforward layers, and select GeLU [17] as the activation function of the feedforward network. Adam [10] optimizer with an initial learning rate of 0.001 was applied. The dropout strategy [18] is adopted to avoid overfitting and the dropout rate is set to 0.2. We use a batch size of 32. The model is trained for 30 epochs to minimize the cross-entropy loss. We release the source code of DMSO and hope to facilitate future research.

https://github.com/anonymousseke/dmso

E. Result

RQ1: How effective is DMSO in predicting knowledge unit pairs of different classes?

Table I presents the experiment results. The best results are highlighted in bold. We can see that the results for the *direct* and *indirect* classes are much lower than the *duplicate* and *isolated* classes, indicating that, the semantic information in these two classes is more confusing for understanding. So that SOFTSVM and DOTBILSTM are difficult to identify patterns for classification, which leads to poor results (34.7%, 50.6%, 55.2%, and 61.0%). This result reflects that this class limits the overall performance of all approaches included DMSO.

Meanwhile, compared to feature-based SOFTSVM, deep learning methods do not rely on fixed patterns and allow advanced abstraction modeling of text to understand the relevance of two knowledge unit pairs and obtain better scores. We can see that DMSO achieves the best results for all four classes, with 83.4%, 83.2%, and 83.3% in overall Precision, Recall, and F1-score. In addition, compared to the state-of-the-art model ASIM, DMSO improves by 1.5% in F1-score, and by 1.3% and 1.5% in both Precision and Recall. This result shows that extracting semantic and interaction features of sentences is important for understanding and identifying the relevance of knowledge unit pairs, especially in long text sequences and complex expressions in software domains.

RQ2: How much influence do the modules we proposed contribute to the improvement of DMSO?

The experimental results are shown in Table II. After removing GIF or DA, respectively, the revised models' performance decreases to some degree (1.7% and 1.0%, in terms of F1-score, respectively). However, after removing the IA or IFE, the model's performance decreases more obviously (2.7% and 4.5%, in terms of F1-score, respectively), especially in predicting the *direct* class (10.3% and 2.5% in terms of F1-score, respectively). This experiment's results prove the importance of the Interaction Feature Extractor and the Global Interaction Fusion Layer, which plays an essential role in predicting question relatedness.

RQ3: Does DMSO work well in other semantic relevance tasks from software engineering domains?

Table III presents the results. DMSO outperforms other models, including rule-based approaches (Jcrd [12]), classifiers (SVM-bas [14], SVM-adv [15], and SOFTSVM), and neural networks (CNN [14], DNN [13], DCNN [15], DOTBILSTM, and ASIM), achieve an accuracy of 97.69%. Moreover, DCNN is the best model in the paper [15] on the AskUbuntu dataset, and the ASIM is the state-of-the-art model on the Knowledge Unit dataset. The experimental results show that DMSO has a degree of generalization capability that can be applied to other software engineering fields.

V. Conclusion and Future Work

In this paper, we propose a deep learning model DMSO we introduce a deep learning model (DMSO) to predict semantic relatedness between questions in Stack Overflow, helping developers acquire related knowledge to solve the programming

Table I
Comparing the result(Precision, Recall, and F1-score) of DMSO and the baseline models in each relatedness class.

	Model/Classes	**Duplicate**	**Direct**	**Indirect**	**Isolated**	**Overall**
Precision	SOFTSVM	0.532	0.497	0.476	0.766	0.568
	DOTBILSTM	0.929	0.561	0.623	0.894	0.752
	ASIM	0.948	0.720	0.698	0.917	0.821
	DMSO	**0.956**	**0.732**	**0.717**	**0.933**	**0.834**
Recall	SOFTSVM	0.584	0.266	0.541	0.843	0.559
	DOTBILSTM	0.902	0.544	0.598	0.911	0.739
	ASIM	0.936	0.712	0.673	0.947	0.817
	DMSO	**0.943**	**0.745**	**0.692**	**0.949**	**0.832**
F1-score	SOFTSVM	0.557	0.347	0.506	0.802	0.553
	DOTBILSTM	0.915	0.552	0.610	0.902	0.745
	ASIM	0.941	0.716	0.685	0.932	0.819
	DMSO	**0.949**	**0.738**	**0.704**	**0.941**	**0.833**

Table II
Comparing the F1-score of DMSO and the revised models

Model/Classes	**Duplicate**	**Direct**	**Indirect**	**Isolated**	**Overall**
DMSO	**0.949**	**0.739**	**0.704**	**0.941**	**0.833**
DMSO-GIF	0.933	0.715	0.679	0.935	0.816
DMSO-IA	0.916	0.714	0.690	0.903	0.806
DMSO-DA	0.938	0.733	0.689	0.932	0.823
DMSO-IFE	0.912	0.636	0.678	0.925	0.788

Table III
Performance of different models in the Askubuntu dataset

Model/Classes	**Accuracy**
Jcrd	0.7291
SVM-bas	0.7025
SVM-adv	0.7587
CNN	0.7450
DNN	0.7865
DCNN	0.7900
DOTBILSTM	0.87
SOFTSVM	0.90
ASIM	0.9625
DMSO	0.9769

issues. The experiment results show DMSO's effectiveness and consistency in predicting question relatedness, outperforming some baseline models in previous works. Moreover, in the duplicate question detection task of AskUbuntu, DMSO can also achieve state-of-the-art performance, which proves its generalization ability. We will explore the implementation of an application that provides related questions when users retrieve questions or post new issues in future work.

References

[1] B. Xu, D. Ye, Z. Xing, X. Xia, G. Chen, and S. Li, "Predicting semantically linkable knowledge in developer online forums via convolutional neural network," in *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2016, pp. 51–62.

[2] Tay Y, Luu A T, Zhang A, et al. Compositional de-attention networks[J]. Advances in Neural Information Processing Systems, 2019, 32.

[3] J. Pei, Y. Wu, Z. Qin, Y. Cong and J. Guan, "Attention-based model for predicting question relatedness on Stack Overflow," in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, 2021, pp. 97-107.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997. III-D

[5] Yang Z, Yang D, Dyer C, et al. Hierarchical attention networks for document classification[C]//Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies. 2016: 1480-1489.

[6] S. Amirreza, X. Bowen, L. David, T. Solorio, and A. Alipour, "Question relatedness on stack overflow: the task, dataset, and corpus-inspired models," in *Proceedings of the AAAI Reasoning for Complex Question Answering Workshop*, 2019.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[8] W. Fu and T. Menzies, "Easy over hard: A case study on deep learning,"in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, 2017, pp. 49–60.

[9] B. Xu, A. Shirani, D. Lo, and M. A. Alipour, "Prediction of relatedness in stack overflow: deep learning vs. svm: a reproducibility study," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2018, pp.1–10.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[11] I. Srba and M. Bielikova, "Why is Stack Overflow Failing? Preserving Sustainability in Community Question Answering," in *IEEE Software*, vol. 33, no. 4, pp. 80-89, 2016.

[12] Y. Wu, Q. Zhang, and X.-J. Huang, "Efficient near-duplicate detection for q&a forum," in *Proceedings of 5th International Joint Conference on Natural Language Processing*, 2011, pp. 1001–1009.

[13] 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.

[14] D. Bogdanova, C. dos Santos, L. Barbosa, and B. Zadrozny, "Detecting semantically equivalent questions in online user forums," in *Proceedings of the 19th Conference on Computational Natural Language Learning*, 2015, pp. 123–131.

[15] J. Rodrigues, C. Saedi, V. Maraev, J. Silva, and A. Branco, "Ways of asking and replying in duplicate question detection," in *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, 2017, pp. 262–270.

[16] V. Efstathiou, C. Chatzilenas, and D. Spinellis, "Word embeddings for the software engineering domain," in *Proceedings of the 15th International Conference on Mining Software Repositories*, 2018, pp.38–41.

[17] D. Hendrycks, and K. Gimpel. "Bridging nonlinearities and stochastic regularizers with gaussian error linear units," *arXiv preprint arXiv: 1606.08415*, 2016.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[19] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473, 2014. III-E