Explicit Interaction Model towards Text Classification

Cunxiao Du,[1] **Zhaozheng Chen,**[1*] **Fuli Feng,**[2*] **Lei Zhu,**[3] **Tian Gan,**[1†] **Liqiang Nie**[1]
[1]Shandong University, No.72 Binhai Road, Jimo,Qingdao, Shandong, China 266237
[2]National University of Singapore, 13 Computing Drive, Singapore 117417
[3]Shandong Normal University, No.1 University Road, Changqing Dist., Ji'nan, Shandong, China 250358
{cnsdunm,zhaozhengcc,fulifeng93,leizhu0608}@gmail.com, {gantian, nieliqiang}@sdu.edu.cn

Abstract

Text classification is one of the fundamental tasks in natural language processing. Recently, deep neural networks have achieved promising performance in the text classification task compared to shallow models. Despite of the significance of deep models, they ignore the fine-grained (matching signals between words and classes) classification clues since their classifications mainly rely on the text-level representations. To address this problem, we introduce the interaction mechanism to incorporate word-level matching signals into the text classification task. In particular, we design a novel framework, EXplicit interAction Model (dubbed as EXAM), equipped with the interaction mechanism. We justified the proposed approach on several benchmark datasets including both multi-label and multi-class text classification tasks. Extensive experimental results demonstrate the superiority of the proposed method. As a byproduct, we have released the codes and parameter settings to facilitate other researches.

Introduction

Text classification is one of the fundamental tasks in natural language processing, targeting at classifying a piece of text content into one or multiple categories. According to the number of desired categories, text classification can be divided into two groups, namely, *multi-label* (multiple categories) and *multi-class* (unique category). For instance, classifying an article into different topics (*e.g.,* machine learning or data mining) falls into the former one since an article could be under several topics simultaneously. By contrast, classifying a comment of a movie into its corresponding rating level lies into the multi-class group. Both multi-label and multi-class text classifications have been widely applied in many fields like sentimental analysis (Cambria, Olsher, and Rajagopal 2014), topic tagging (Grave et al. 2017), and document classification (Yang et al. 2016).

Feature engineering dominates the performance of traditional shallow text classification methods for a very long time. Various rule-based and statistical features like bag-of-words (Wallach 2006) and N-grams (Brown et al. 1992) are designed to describe the text, and fed into the shallow machine learning models such as Linear Regression (Zhu and Hastie 2001) and Support Vector Machine (Cortes and Vapnik 1995) to make the judgment. Traditional solutions suffer from two defects: 1) High labor intensity for the manually crafted features, and 2) data sparsity (a N-grams could occur only several times in a given dataset).

Recently, owing to the ability of tackling the aforementioned problems, deep neural networks (Kim 2014; Iyyer et al. 2015; Schwenk et al. 2017; Liu, Qiu, and Huang 2016; Grave et al. 2017) have become the promising solutions for the text classification. Deep neural networks typically learn a *word-level representation* for the input text, which is usually a matrix with each row/column as an embedding of a word in the text. They then compress the word-level representation into a *text-level representation* (vector) with aggregation operations (*e.g.,* pooling). Thereafter, a fully-connected (FC) layer at the topmost of the network is appended to make the final decision. Note that these solutions are also called *encoding-based methods* (Munkhdalai and Yu 2017), since they encode the textual content into a latent vector representation.

Although great success has been achieved, these deep neural network based solutions naturally ignore the fine-grained classification clues (*i.e.,* matching signals between words and classes), since their classifications are based on text-level representations. As shown in Figure 1, the classification (*i.e.,* FC) layer of these solutions matches the text-level representation with class representations via a dot-product operation. Mathematically, it interprets the parameter matrix of the FC layer as a set of class representations (each column is associated with a class) (Press and Wolf 2017). As such, the probability of the text belonging to a class is largely determined by their overall matching score regardless of word-level matching signals, which would provide explicit signals for classification (*e.g., missile* strongly indicates the topic of *military*).

To address the aforementioned problems, we introduce the interaction mechanism (Wang and Jiang 2016b), which is capable of incorporating the word-level matching signals for text classification. The key idea behind the interaction mechanism is to explicitly calculate the matching scores between the words and classes. From the word-level representation, it computes an interaction matrix, in which each entry is the matching score between a word and a class (dot-product

*Equal contribution.
†Corresponding Author.

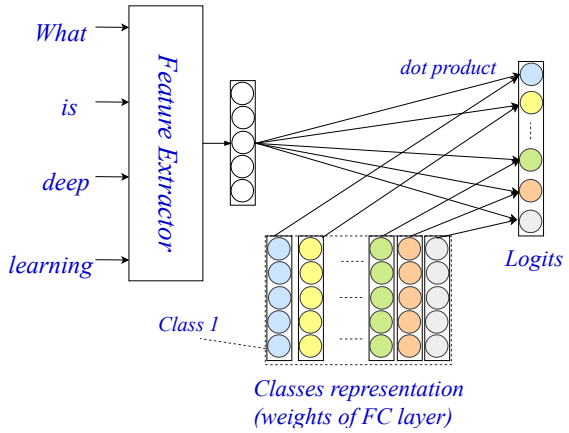

Figure 1: Illustration of encoding-based methods for text classification with text-level matching.

between their representations), illustrating the word-level matching signals. By taking the interaction matrix as a text representation, the later classification layer could incorporate fine-grained word level signals for the finer classification rather than simply making the text-level matching.

Based upon the interaction mechanism, we devise an EXplicit interAction Model (dubbed as *EXAM*). Specifically, the proposed framework consists of three main components: *word-level encoder*, *interaction layer*, and *aggregation layer*. The word-level encoder projects the textual contents into the word-level representations. Hereafter, the interaction layer calculates the matching scores between the words and classes (*i.e.,* constructs the interaction matrix). Then, the last layer aggregates those matching scores into predictions over each class, respectively. We justify our proposed EXAM model over both the multi-label and multi-class text classifications. Extensive experiments on several benchmarks demonstrate the effectiveness of the proposed method, surpassing the corresponding state-of-the-art methods remarkably.

In summary, the contributions of this work are threefold:

- We present a novel framework, EXAM, which leverages the interaction mechanism to explicitly compute the word-level interaction signals for the text classification.
- We justify the proposed EXAM model over both multi-label and multi-class text classifications. Extensive experimental results demonstrate the effectiveness of the proposed method.
- We release the implementation of our method (including some baselines) and the involved parameter settings to facilitate later researchers[1].

Preliminaries

In this section, we introduce two widely-used word-level encoders: *Gated Recurrent Units* (Chung et al. 2014) and *Region Embedding* (Qiao et al. 2018). These encoders project a piece of input text into a word-level representation, serving as the building blocks of the proposed method. For the notations in this paper, we use bold capital letters (e.g., $\mathbf{X}$) and bold lowercase letters (e.g., $\mathbf{x}$) to denote matrices and vectors, respectively. We employ non-bold letters (e.g., x) to represent scalars, and Greek letters (e.g., α) as parameters. $\mathbf{X_{i,:}}$ is used to refer the i-th row of the matrix $\mathbf{X}$, $\mathbf{X_{:,j}}$ to represent the j-th column vector and $\mathbf{X_{i,j}}$ to denote the element in the i-th row and j-th column.

Gated Recurrent Units

Owing to the ability of capturing the sequential dependencies and being easily optimized (*i.e.,* avoid the gradient vanishing and explosion problems), Gated Recurrent Units (GRU) becomes a widely used word-level encoder (Liu, Qiu, and Huang 2016; Yogatama et al. 2017). Typically, a GRU generates word-level representations in two phases: 1) mapping each word in the text into an embedding (a real-valued vector), and 2) projecting the sequence of word embeddings into a sequence of hidden representations, which encodes the sequential dependencies.

Word embedding. Word embedding is a general method to map a word from one hot vector to a low dimensional and real-valued vector. With enough data, word embedding can capture high-level representations of words.

Hidden representation. Given an embedding feature sequence $\mathbf{E} = [\mathbf{E_{1,:}}, \mathbf{E_{2,:}}, \cdots, \mathbf{E_{n,:}}]$, GRU will compute a vector $\mathbf{H_{i,:}}$ at the i-th time-step for each $\mathbf{E_{i,:}}$, and $\mathbf{H_{i,:}}$ is defined as:

$$\begin{cases} \mathbf{r_i} = \sigma(\mathbf{M_r} \cdot [\mathbf{H_{i-1,:}}, \mathbf{E_{i,:}}]), \\ \mathbf{z_i} = \sigma(\mathbf{M_z} \cdot [\mathbf{H_{i-1,:}}, \mathbf{E_{i,:}}]), \\ \widetilde{\mathbf{H_{i,:}}} = \mathbf{tanh}(\mathbf{M_r} \cdot [\mathbf{H_{i-1,:}}, \mathbf{E_{i,:}}]), \\ \mathbf{H_{i,:}} = (\mathbf{1} - \mathbf{z_i}) * \mathbf{H_{i-1,:}} + \mathbf{z_i} * \widetilde{\mathbf{H_{i,:}}}, \end{cases} \tag{1}$$

where $\mathbf{M_r}$ and $\mathbf{M_z}$ are trainable parameters in the GRU, and σ and $tanh$ are sigmoid and tanh activation functions, respectively. The sequence of hidden representations $\mathbf{H} = [\mathbf{H_{1,:}}, \cdots, \mathbf{H_{n,:}}]$ is denoted as the word-level representation of the input text.

Region Embedding

Although word embedding is a good representation for the word, it can only compute the feature vector for the single word. Qiao et al. (2018) proposed region embedding to learn and utilize task-specific distributed representations of N-grams. In the region embedding layer, the representation of a word has two parts, the embedding of the word itself and a weighting matrix to interact with the local context. For the word w_i, the first part $\mathbf{e_{w_i}}$ is learned by an embedding matrix $\mathbf{E} \in \mathbb{R}^{k \times v}$ and the second part $\mathbf{K_{w_i}} \in \mathbb{R}^{k \times (2 \times s+1)}$ is looked up in the tensor $\mathbf{U} \in \mathbb{R}^{k \times (2 \times s+1) \times v}$ by w_i's index in the vocabulary, where v is the size of the vocabulary, $2 \times s + 1$ the region size and k the embedding size. And then, each column in $\mathbf{K_{w_i}}$ is used to interact with the context word in the corresponding relative position of w_i to get the context-aware $\mathbf{p^t_{w_{i+t}}}$ for each word w_{i+t} in the region. Formally it is computed by the following function:

$$\mathbf{p^i_{w_{i+t}}} = \mathbf{K_{w_i,t}} \odot \mathbf{e_{w_{i+t}}}, \tag{2}$$

[1] https://github.com/NonvolatileMemory/AAAI_2019_EXAM .

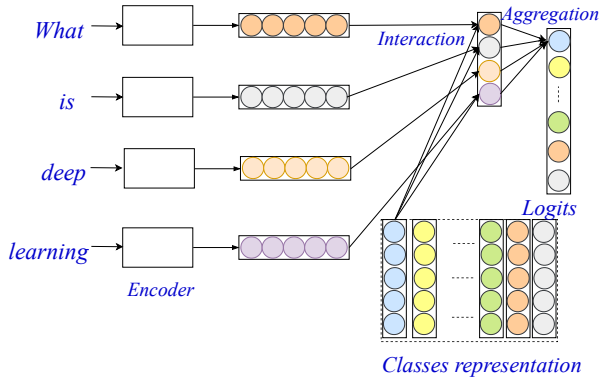

Figure 2: Illustration of proposed EXAM method with word-level matching.

where $\odot$ denotes element-wise multiply. And the final representation $\mathbf{r_{i,s}}$ of the middle word w_i is computed as follows:

$$\mathbf{r_{i,s}} = max([\mathbf{p^i_{w_{i-s}}}, \mathbf{p^i_{w_{i-s+1}}}, \cdots, \mathbf{p^i_{w_{i+s-1}}}, \mathbf{p^i_{w_{i+s}}}]). \quad (3)$$

Model

Problem Formulation

- Multi-Class Classification. In this task, we should categorize each text instance to precisely one of c classes. Suppose that we have a data set $\mathcal{D} = \{d_i, \mathbf{l_i}\}^N$, where d_i denotes the text and the one-hot vector $\mathbf{l_i} \in \mathbb{R}^c$ represents the label for d_i, our goal is to learn a neural network $\mathcal{N}$ to classify the text.
- Multi-Label Classification. In this task, each text instance belongs to a set of c target labels. Formally, suppose that we have a dataset $\mathcal{D} = \{d_i, \mathbf{l_i}\}_{i=1}^N$, where d_i denotes the text and the multi-hot vector $\mathbf{l_i}$ represents the label for the text d_i. Our goal is to learn a neural network $\mathcal{N}$ to classify the text.

Model Overview

Motivated by the limitation of encoding-based models for text classification, which is lacking the fine-grained classification clue, we propose a novel framework, named *EXplicit interAction Model* (EXAM), leveraging the interaction mechanism to incorporate word-level matching signals. As can be seen from Figure 2, EXAM mainly contains three components:

- A *word-level encoder* to project the input text d_i into a word-level representation $\mathbf{H}$.
- An *interaction layer* to compute the interaction signals between the words and classes.
- An *aggregation layer* to aggregate the interaction signals for each class and make the final predictions.

Considering that word-level encoders are well investigated in previous studies (as mentioned in the Section 2), and the target of this work is to learn the fine-grained classification signals, we only elaborate the interaction layer and aggregation layer in the following subsections.

Interaction Layer

Interaction mechanism is widely used in tasks of matching source and target textual contents, such as natural language inference (Wang and Jiang 2016b) and retrieve-based chatbot (Wu et al. 2017). The key idea of interaction mechanism is to use the interaction features between the small units (*e.g.,* words in the textual contents) to infer fine-grained clues whether two contents are matching. Inspired by the success of methods equipped with interaction mechanism over encode-based methods in matching the textual contents, we introduce the interaction mechanism into the task of matching textual contents with their classes (*i.e.,* text classification).

Specifically, we devise an interaction layer which aims to compute the matching score between the word and class. Different from conventional interaction layer, where the word-level representations of both source and target are extracted with encoders like GRU, here we first project classes into real-valued latent representations. In other words, we employ a trainable representation matrix $\mathbf{T} \in \mathbb{R}^{c \times k}$ to encode classes (each row represents a class), where c denotes the amount of classes and k is the embedding size equals to that of words. We then adopt *dot product* as the *interaction function* to estimate the matching score between the target word t and class s, of which the formulation is,

$$\mathbf{I_{st}} = \mathbf{T_{s,:}}\mathbf{H^T_{t,:}}, \quad (4)$$

where $\mathbf{H} \in \mathbb{R}^{n \times k}$ denotes word-level representation of the text, extracted by the encoder with n denoting the length of the text. In this way, we can compute the interaction matrix $\mathbf{I} \in \mathbb{R}^{c \times n}$ by following:

$$\mathbf{I} = \mathbf{T}\mathbf{H}^\mathbf{T}. \quad (5)$$

Note that we reject more complex interaction functions like *element-wise multiply* (Gong, Luo, and Zhang 2017) and *cosine similarity* (Wang, Hamza, and Florian 2017) for the consideration of efficiency.

Aggregation Layer

This layer is devised to aggregate the interaction features for each class s into a logits o_i^s, which denotes the matching score between class s and the input text d_i. The aggregation layer can be implemented in different ways such as CNN (Gong, Luo, and Zhang 2017) and LSTM (Wang, Hamza, and Florian 2017). However, to keep the simplicity and efficiency of EXAM, here we only use a MLP with two FC layers, where ReLU is employed as the activation function of the first layer. Formally, the MLP aggregates the interaction features $\mathbf{I_{s,:}}$ for class s, and compute its associated logits as following:

$$\begin{cases} \mathbf{A_{s,:}} = ReLU(\mathbf{I_{s,:}}\mathbf{W_1} + \mathbf{b}), \\ o_i^s = \mathbf{A_{s,:}}\mathbf{W_2}, \end{cases} \quad (6)$$

where $\mathbf{W_1}$ and $\mathbf{W_2}$ are trainable parameters and $\mathbf{b}$ is the bias in the first layer.

We then normalize the logits $\mathbf{o_i} = [o_i^1, \cdots, o_i^c]$ into probabilities $\mathbf{p_i}$. Note that we follow previous work (Grave et al. 2017) and employ *softmax* and *sigmoid* for multi-class and multi-label classifications, respectively.

Table 1: Statistics of Datasets.

Dataset	Classes	Average Lengths	Train Samples	Test Samples	Tasks
Amazon Review Polarity	2	91	3,600,000	400,000	Sentiment
Amazon Review Full	5	93	3,000,000	650,000	Analysis
AG's News	4	44	120,000	7,600	News Classification
Yahoo! Answers	10	112	1,400,000	60,000	Question Answer
DBPedia	14	55	560,000	70,000	Ontology Extraction

Loss Function

Similar to previous studies (Schwenk et al. 2017), in the multi-class text classification, we use cross entorpy loss as our loss function:

$$\mathcal{L}_{loss} = -\sum_{i=1}^{N}\sum_{j=1}^{c}(l_i^j log(p_i^j)). \quad (7)$$

Following previous researchers (Grave et al. 2017), we choose binary classification loss as our loss function for the multi-label one:

$$\mathcal{L}_{loss} = -\sum_{i=1}^{N}\sum_{j=1}^{c}(l_i^j log(p_i^j) + (1-l_i^j) log(1-p_i^j)). \quad (8)$$

Generalized Encoding-Based Model

In this section, we elaborate how the *encoding-based model* can be interpreted as a special case of our EXAM framework. As FastText (Grave et al. 2017) is the most popular model for text classification and has been investigated extensively in the literature, being able to recover it allows EXAM to mimic a large family of text classification models.

FastText contains three layers: 1) an embedding layer to get the word-level representation $\mathbf{H}_{\mathbf{t},:}$ for the word t, 2) an average pooling layer to get the text-level representation $\mathbf{f} \in \mathbb{R}^{1\times k}$, and 3) a FC layer to get the final logits $\mathbf{p} \in \mathbb{R}^{1\times c}$, where k denotes the embedding size and c means the number of classes. Note that we omit the subscript of the document ID for conciseness. Formally, it computes the logits p^s of s-th class as follows:

$$\begin{cases} \mathbf{f} = \frac{1}{n}\sum_{t=1}^{n}\mathbf{H}_{\mathbf{t},:}, \\ p^s = \mathbf{f}\mathbf{W}_{:,\mathbf{s}} + \mathbf{b}_{\mathbf{s}}, \end{cases} \quad (9)$$

where $\mathbf{W} \in \mathbb{R}^{k\times c}$ and $\mathbf{b} \in \mathbb{R}^{1\times c}$ are the trainable parameters in the last FC layer, and n denotes the length of the text. The Eqn.(9) has an equivalent form as following:

$$p^s = \frac{1}{n}\sum_{t=1}^{n}(\mathbf{H}_{\mathbf{t},:}\mathbf{W}_{:,\mathbf{s}}) + \mathbf{b}_{\mathbf{s}}. \quad (10)$$

It is worth noting that $\mathbf{H}_{\mathbf{t},:}\mathbf{W}_{:,\mathbf{s}}$ is exactly the interaction feature between word t and class s. Therefore, the FastText is a special case of EXAM with an *average pooling* as the aggregation layer. In EXAM, we use a non-linear MLP to be the aggregation layer, and it will generalize FastText to a non-linear setting which might be more expressive than the original one.

Experiments

Multi-Class Classification

Datasets We used publicly available benchmark datasets from (Zhang, Zhao, and LeCun 2015) to evaluate EXAM. There are in total 6 text classification datasets, corresponding to sentiment analysis, news classification, question-answer and ontology extraction tasks, respectively. Table 1 shows the descriptive statistics of datasets used in our experiments. Stanford tokenizer is used to tokenize the text and all words are converted to lower case. We used padding to handle the various lengths of the text, and different maximum lengths are set for each dataset, respectively. If the length of the text is less than the corresponding predefined value, we padded it with zero; otherwise we truncated the original text. To guarantee a fair comparison, the same evaluation protocol of (Zhang, Zhao, and LeCun 2015) is employed. We split 10% samples from the training set as the validation set to perform early stop for our models.

Hyperparameters For the multi-class task, we chose region embedding as the Encoder in EXAM. The region size is 7 and embedding size is 128. We used adam (Kingma and Ba 2014) as the optimizer with the initial learning rate 0.0001 and the batch size is set to 16. As for the aggregation MLP, we set the size of the hidden layer as 2 times interaction feature length. Our models are implemented and trained by MXNet (Chen et al. 2015) with a single NVIDIA TITAN Xp.

Baselines To demonstrate the effectiveness of our proposed EXAM, we compared it with several state-of-the-art baselines. The baselines are mainly in three variants: 1) models based on feature engineering; 2) Char-based deep models, and 3) Word-based deep models. The first category uses the feature from the text to conduct the classification, and we reported the results from BoW (Zhang, Zhao, and LeCun 2015), N-grams (Zhang, Zhao, and LeCun 2015) and N-grams TFIDF (Zhang, Zhao, and LeCun 2015) as baselines. The second one means the input of the model is the character in the original text, and we chose the Char-CNN (Zhang, Zhao, and LeCun 2015), Char-CRNN (Zhang, Zhao, and LeCun 2015) and VDCNN (Schwenk et al. 2017) as baselines. As for the word-based deep models, the text is pre-segmented into words as the input, and we applied Small word CNN (Zhang, Zhao, and LeCun 2015), Large word CNN (Zhang, Zhao, and LeCun 2015), LSTM (Zhang, Zhao, and LeCun 2015), FastText (Grave et al. 2017) and W.C RegionEmb (Qiao et al. 2018) as the baselines. It is worth emphasizing

Table 2: Test Set Accuracy [%] on multi-class document classification tasks, and all the results of baselines are directly cited from the respective papers. The three different models are separated by lines. The best scores for the category are marked with underline and the overall best scores are highlight with bold font.

Model	Amz. P.	Amz. F.	AG	Yah. A.	DBP
BoW (Zhang, Zhao, and LeCun 2015)	90.4	54.6	88.8	68.9	96.6
N-grams (Zhang, Zhao, and LeCun 2015)	92.0	54.3	92.0	68.5	98.6
N-grams TFIDF (Zhang, Zhao, and LeCun 2015)	91.5	52.4	92.4	68.5	98.7
Char-CNN (Zhang, Zhao, and LeCun 2015)	94.5	59.6	87.2	71.2	98.3
Char-CRNN (Zhang, Zhao, and LeCun 2015)	94.1	59.2	91.4	71.7	98.6
VDCNN (Schwenk et al. 2017)	**95.7**	**63.0**	91.3	73.4	98.7
Small word CNN (Zhang, Zhao, and LeCun 2015)	94.2	56.3	89.1	70.0	98.2
Large word CNN (Zhang, Zhao, and LeCun 2015)	94.2	54.1	91.5	71.0	98.3
LSTM (Zhang, Zhao, and LeCun 2015)	93.9	59.4	86.1	70.8	98.6
Bigram-FastText (Grave et al. 2017)	94.6	60.2	92.5	72.3	98.6
W.C RegionEmb (Qiao et al. 2018)	95.1	60.9	92.8	73.7	98.9
EXAM (Ours)	95.5	61.9	**93.0**	**74.8**	**99.0**

that all the baselines and our EXAM do not use pre-trained word embedding over other corpus like glove.

Overall Performance We compared our EXAM to several state-of-the-art baselines with respect to accuracy. All results are summarized in Table 1. Four points are observed as following:

- Models based on feature engineering get the worst results on all the five datasets compared to the other methods. The main reason is that the feature engineering cannot take full advantage of the supervision from the training set and it also suffers from the data sparsity.
- Char-based models get the highest overall scores on the two Amazon datasets. There are possibly two reasons, 1) compared to the word-based models, char-based models enrich the supervision from characters and the characters are combined to form N-grams, stems, words and phrase which are helpful in the sentimental classification. 2) The two Amazon datasets contain millions of training samples, perfectly fitting the deep residual architecture for the VDCNN. For the three char-based baselines, VDCNN gets the best performance on almost all the datasets because it has 29 convolutional layers allowing the model to learn more combinations of characters.
- Word-based baselines exceed the other variants on three datasets and lose on the two Amazon datasets. The main reason is that the three tasks like news classification conduct categorization mainly via key words, and the word-based models are able to directly use the word embedding without combining the characters. For the five baselines, W.C RegionEmb performs the best, because it learns the region embedding to utilize the N-grams feature from the text.
- It is clear to see that EXAM achieves the best performance over the three datasets: AG, Yah. A. and DBP. For the Yah.A., EXAM improves the best performance by 1.1%. Additionally, as a word-based model, EXAM beats all the word-based baselines on the other two Amazon datasets with a performance gain of 1.0% on the Amazon Full, because our EXAM considers more fine-grained interaction features between classes and words, which is quite helpful in this task.

Table 3: Component-wise evaluation.

Dataset	EXAM	EXAM$_{Encoder}$
Amz. P.	**95.5**	95.1
Amz. F.	**61.9**	60.9
AG	**93.0**	92.8
Yah. A.	**74.8**	73.1
DBP	**99.0**	98.9

Component-wise Evaluation We studied the variant of our model to further investigate the effectiveness of the interaction layer and aggregation layer. We built a model called EXAM$_{Encoder}$ to preserve only the Encoder component with a max pooling layer and FC layer to derive the final probabilities. EXAM$_{Encoder}$ does not consider the interaction features between the classes and words, so it will automatically be degenerated into the Encoding-Based model. We reported the results of the two models on all the datasets at Table 3, and it is clear to see that EXAM$_{Encoder}$ is not a patch on the original EXAM, verifying the effectiveness of interaction mechanism. We also drew the convergence lines for EXAM and the EXAM$_{Encoder}$ for the datasets. From the Figure 3, where the red lines represent EXAM and the blue is EXAM$_{Encoder}$, we observed that EXAM converges faster than EXAM$_{Encoder}$ with respect to all the datasets. Therefore, the interaction brings not only performance improvement but also faster convergence. The possible reason is that a non-linear aggregation layer introduces more parameters to fit the interaction features compared to the average pooling layer as mentioned in Section 4.

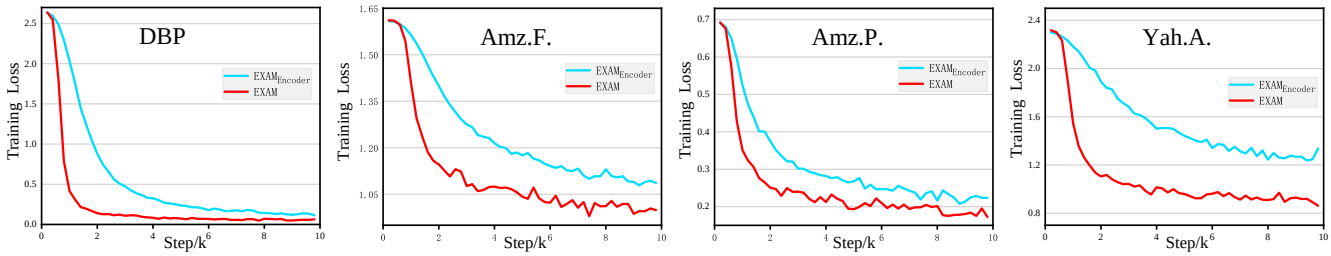

Figure 3: Convergence lines on the four dataset DBP, Amz. F., Amz. P. and Yah. A., respectively.

Table 4: Performance comparison between EXAM and baselines. The best scores are highlight in bold font.

	Kanshan-Cup Dataset			Zhihu Dataset		
Model	Precision	Recall@5	F_1	Precision	Recall@5	F_1
Char-CNN (Zhang, Zhao, and LeCun 2015)	1.299	0.536	0.379	-	-	-
Char-TextRNN (Liu, Qiu, and Huang 2016)	1.304	0.537	0.380	-	-	-
FastText (Grave et al. 2017)	1.325	0.546	0.387	1.235	0.564	0.387
TextCNN (Kim 2014)	1.331	0.550	0.389	1.241	0.566	0.389
Word-TextRNN (Liu, Qiu, and Huang 2016)	1.345	0.555	0.393	1.240	0.566	0.389
EXAM (Ours)	**1.360**	**0.561**	**0.397**	**1.267**	**0.578**	**0.397**

Multi-Label Classification

Datasets We conducted experiments on two different multi-label text classification datasets, named KanShan-Cup dataset[2] (a benchmark) and Zhihu dataset[3], respectively.

- **KanShan-Cup dataset**. This dataset is released by a competition of tagging topics for questions (*multi-label classification*) posted in the largest Chinese community question answering platform, Zhihu. The dataset contains 3,000,000 questions and 1,999 topics (classes), where one question may belong to one to five topics. For questions with more than 30 words, we kept the last 30 words, otherwise, we padded zeros. We separated the dataset into training, validation, and testing with 2,800,000, 20,000, and 180,000 questions, respectively.
- **Zhihu dataset**. Considering the user privacy and data security, KanShan-Cup does not provide the original texts of the questions and topics, but uses numbered codes and numbered segmented words to represent text messages. Therefore, it is inconvenient for researchers to perform analyses like visualization and case study. To solve this problem, we constructed a dataset named Zhihu dataset. We chose the top 1,999 frequent topics from Zhihu and crawled all the questions relevant to these topics. Finally, we acquired 3,300,000 questions, with less than 5 topics for each question. We adopted 3,000,000 samples as the training set, 30,000 samples as validation and 300,000 samples as testing.

Baselines We applied the following models as baselines to evaluate the effectiveness of EXAM.

- **Char-based Model**. We chose Char-CNN (Zhang, Zhao, and LeCun 2015) and Char-RNN (Liu, Qiu, and Huang 2016) as the baselines to represent this kind of methods.
- **Word-based Model**. For the word-based models, we reported the results from TextCNN (Kim 2014), TextRNN (Liu, Qiu, and Huang 2016) and FastText (Grave et al. 2017). The three models got the best performance in the KanShan-Cup competition, so we applied them as the word-based baselines.

Hyperparameters We implemented the baseline models and EXAM by MXNet (Chen et al. 2015). We used the matrix trained by word2vec (Mikolov et al. 2013) to initialize the embedding layer, and the embedding size is 256. We adopted GRU as the Encoder, and each GRU Cell has 1,024 hidden states. The accumulated MLP has 60 hidden units. We applied Adam (Kingma and Ba 2014) to optimize models on one NVIDIA TITAN Xp with the batch size of 1000 and the initial learning rate is 0.001. The validation set is applied for early-stopping to avoid overfitting. All hyperparameters are chosen empirically.

Metrics We used the following metrics to evaluate the performance of our model and baseline models.

- Precision: Different from the traditional precision metric (Precision@5) which is set as the fraction of the relevant topic tags among the five returned tags, we utilized weighted precision to encourage the relevant topic tags to be ranked higher in the returned list. Formally, the Precision is computed as following,

$$Precision = \sum_{pos \in \{1,2,3,4,5\}} \frac{Precision@pos}{log(pos+1)}. \quad (11)$$

[2] https://biendata.com/competition/zhihu/.
[3] www.zhihu.com.

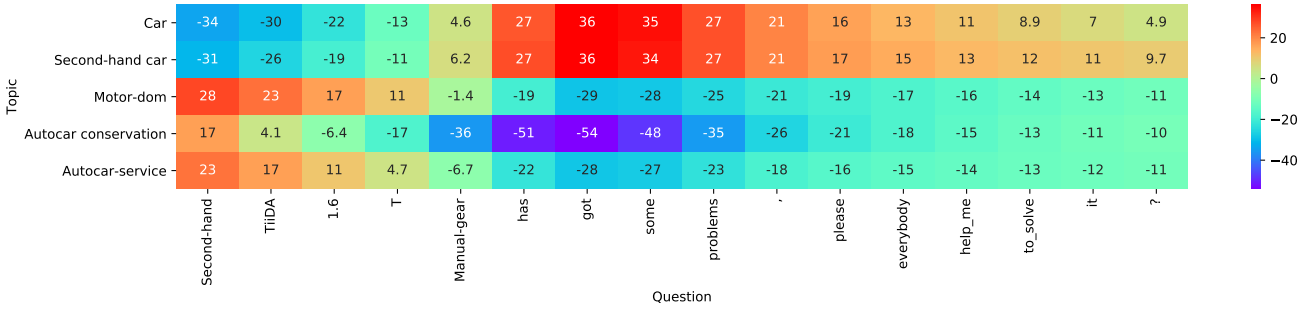

Figure 4: The visualization of interaction features of EXAM.

- Recall@5: Recall is the fraction of relevant topic tags that have been retrieved over the total amount of five relevant topic tags, high recall means that the model returns most of the relevant topic tags.
- F_1: F_1 is the harmonic average of the precision and recall, we computed it as following,

$$F_1 = \frac{Precision * Recall@5}{Precision + Recall@5}. \quad (12)$$

Performance Comparison Table 4 gives the performance of our model and baselines over two different datasets with respect to Precision, Recall@5 and F_1. We observed the following from the Table 4:

- Word-based models are better than char-based models in Kanshan-Cup dataset. That may be because in Chinese the words can offer more supervisions than characters and the question tagging task needs more word supervision.
- For word-based baseline models, all the baselines have similar performance which corroborates the conclusion in FastText (Grave et al. 2017) that simple network is on par with deep learning classifiers in text classification.
- Our models achieve the state-of-the-art performance over two different datasets though we only slightly modified TextRNN to build EXAM. Different from the traditional models which encode the whole text into a vector, in EXAM, the representations of classes firstly interact with words to get more fine-grained features as shown in Figure 4. The results suggest that word-level interaction features are relatively more important than global text-level representations in this task.

Interaction Visualization To illustrate the effectiveness of explicit interaction, we visualized an interaction feature $\mathbf{I}$ of the question "Second-hand TIDDA 1.6 T Mannual gear has gotten some problems, please everybody help me to solve it ?". This question has 5 topics: Car, Second-hand Car, Motor Dom, Autocar Conversation and Autocar Service. EXAM only misclassified the last topic. In Figure 4, we observed that when classifying different topics, the interaction features are different. The topics "Car" and "Second-hand Car" pay much attention to the words like "Second-hand TIIDA" and the other topic like "Autocar Conversation" focuses more on "got some problems". The results clearly signify that the interaction feature between the word and class is well-learned and highly meaningful.

Related Work

Text Classification Existing researches on text classification can be categorized into two groups: feature-based and deep neural models. The former focuses on handcraft features and uses machine learning algorithms as the classifier. Bag-of-words (Wallach 2006) is a very efficient way to conduct the feature engineering. SVM and Naive Bayes are constantly the classifier. The latter, deep neural models, taking advantage of neural networks to accomplish the model learning from data, have become the promising solution for the text classification. For instance, Iyyer et al. (2015) proposed Deep Averaging Networks (DAN) and Grave et al. (2017) proposed the FastText, and both are simple but efficient. To get the temporal features between the words in the text, some models like TextCNN (Kim 2014) and Char-CNN (Zhang, Zhao, and LeCun 2015) exploit the convolutional neural network, and there are also some models based on Recurrent Neural Network (RNN). Recently, Johnson et al. (2017) investigated the residual architecture and built a model called VD-CNN and Qiao et al. (2018) proposed a new method of region embedding for the text classification. However, as mentioned in the Introduction, all these methods are text-level models while EXAM conducts the matching at the word level.

Interaction Mechanism Interaction Mechanism is widely used in Natural Language Sentence Matching (NLSM). The key idea of interaction mechanism is to use the interaction features between the small units (like words in sentence) to make the matching. Wang et al. (2016b) proposed a "matching-aggregation" framework to perform the interaction in Natural Language Inference. Following this work, Parikh et al. (2016) integrated the attention mechanism into this framework, called Decomposable Attention Model. Then Wang et al. (2016a) discussed different interaction functions in Text Matching. Yu et al. (2017) adopted tree-LSTM to get different level units to perform the interaction.Gong et al. (2017) proposed a densely interactive inference network to use DenseNet to aggregate dense interaction features. Our work is different from them since they mainly apply this mechanism in text matching instead of the classification.

Conclusion

In this work, we present a novel framework named EXAM which employs the interaction mechanism to explicitly compute the word-level interaction signals for the text classification. We apply the proposed EXAM on multi-class and multi-label text classifications. Experiments over several benchmark datasets verify the effectiveness of our proposed mechanism. In the future, we plan to investigate the effect of different interaction functions in the interaction mechanism. Besides, we are interested in extend EXAM by introducing more complex aggregation layers like ResNet or DenseNet.

Acknowledgments

This work is supported by the National Basic Research Program of China (973 Program), No.: 2015CB352502; National Natural Science Foundation of China, No.: 61772310, No.:61702300, and No.:61702302; the Project of Thousand Youth Talents 2016; and the Tencent AI Lab Rhino-Bird Joint Research Program (No.JR201805); Fundamental Research Funds of Shandong University (No. 2017HW001).

References

Brown, P. F.; Desouza, P. V.; Mercer, R. L.; Pietra, V. J. D.; and Lai, J. C. 1992. Class-based n-gram models of natural language. *Computational linguistics* (4):467–479.

Cambria, E.; Olsher, D.; and Rajagopal, D. 2014. Senticnet 3: A common and common-sense knowledge base for cognition-driven sentiment analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1515–1521.

Chen, T.; Li, M.; Li, Y.; Lin, M.; Wang, N.; Wang, M.; Xiao, T.; Xu, B.; Zhang, C.; and Zhang, Z. 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR* abs/1512.01274.

Chung, J.; Gülçehre, Ç.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* abs/1412.3555.

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Machine Learning* 20(3):273–297.

Gong, Y.; Luo, H.; and Zhang, J. 2017. Natural language inference over interaction space. *CoRR* abs/1709.04348.

Grave, E.; Mikolov, T.; Joulin, A.; and Bojanowski, P. 2017. Bag of tricks for efficient text classification. In *Proceedings of the European Chapter of the Association for Computational Linguistics*, 427–431.

Iyyer, M.; Manjunatha, V.; Boyd-Graber, J. L.; and III, H. D. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1681–1691.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *CoRR* abs/1408.5882.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Liu, P.; Qiu, X.; and Huang, X. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2873–2879.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 3111–3119.

Munkhdalai, T., and Yu, H. 2017. Neural semantic encoders. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*, 397.

Parikh, A. P.; Täckström, O.; Das, D.; and Uszkoreit, J. 2016. A decomposable attention model for natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2249–2255.

Press, O., and Wolf, L. 2017. Using the output embedding to improve language models. In *Proceedings of the European Chapter of the Association for Computational Linguistics*, 157–163.

Qiao, C.; Huang, B.; Niu, G.; Li, D.; Dong, D.; He, W.; Yu, D.; and Wu, H. 2018. A new method of region embedding for text classification. In *International Conference on Learning Representations*.

Schwenk, H.; Barrault, L.; Conneau, A.; and LeCun, Y. 2017. Very deep convolutional networks for text classification. In *Proceedings of the European Chapter of the Association for Computational Linguistics*, 1107–1116.

Wallach, H. M. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the International Conference on Machine Learning*, 977–984.

Wang, S., and Jiang, J. 2016a. A compare-aggregate model for matching text sequences. *CoRR* abs/1611.01747.

Wang, S., and Jiang, J. 2016b. Learning natural language inference with LSTM. In *The North American Chapter of the Association for Computational Linguistics*, 1442–1451.

Wang, Z.; Hamza, W.; and Florian, R. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 4144–4150.

Wu, Y.; Wu, W.; Xing, C.; Zhou, M.; and Li, Z. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*, 496–505.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A. J.; and Hovy, E. H. 2016. Hierarchical attention networks for document classification. In *The North American Chapter of the Association for Computational Linguistics*, 1480–1489.

Yogatama, D.; Dyer, C.; Ling, W.; and Blunsom, P. 2017. Generative and discriminative text classification with recurrent neural networks. *CoRR* abs/1703.01898.

Yu, H., and Munkhdalai, T. 2017. Neural tree indexers for text understanding. In *Proceedings of the European Chapter of the Association for Computational Linguistics*, 11–21.

Zhang, X.; Zhao, J. J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, 649–657.

Zhu, J., and Hastie, T. 2001. Kernel logistic regression and the import vector machine. In *Advances in Neural Information Processing Systems*, 1081–1088.