Information Processing and Management xxx (xxxx) xxxx

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Pre-train, Interact, Fine-tune: a novel interaction representation for text classification

Jianming Zheng[a], Fei Cai[*,a], Honghui Chen[a], Maarten de Rijke[b]

[a] *Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Hunan, 410073, China*
[b] *Informatics Institute, University of Amsterdam, Amsterdam, 1098 XH, the Netherlands*

ARTICLE INFO

Keywords:
Interaction representation
Pre-training
Fine-tuning
Classification

ABSTRACT

Text representation can aid machines in text understanding. Previous work on text representation often focuses on the so-called forward implication, i.e., preceding words are taken as the context of later words for creating representations, effective it is, yet ignoring the fact that the semantics of a text segment is a product of the mutual implication of words in the text: later words contribute to the meaning of preceding words. To bridge this gap, we introduce the concept of interaction and propose a two-perspective interaction representation, in which it encapsulates a local and a global interaction representation. Here, a *local* interaction representation is one that interacts among words with parent-children relationships on the syntactic trees whereas a *global* interaction interpretation is one that interacts among all the words in a sentence. We combine these two interaction representations to develop a Hybrid Interaction Representation (HIR).

Inspired by existing feature-based and fine-tuning-based pretrain-finetuning approaches to language models, we integrate the merits of feature-based and fine-tuning-based methods to propose the Pre-train, Interact, Fine-tune (PIF) architecture.

We evaluate our proposed models on five widely-used datasets for text classification tasks. It turns out that our ensemble method, HIR$_P$, outperforms state-of-the-art baselines with improvements ranging from 2.03% to 3.15% in terms of error rate. In addition, we find that, the improvements of PIF against most state-of-the-art methods is not affected by increasing of the text length.

1. Introduction

Text representations map text spans into real-valued vectors or matrices (Kenter, Borisov, & de Rijke, 2016). They have come to play a crucial role in machine understanding of text. Applications on that comprises sentiment classification (Tang, Qin, & Liu, 2015), question answering (Qin, Qinmin, Jimmy Xiangji, Liang, & Weijie, 2017), summarization (Ren, Chen, Ren, & de Rijke, 2017), and sentence inference (Parikh, Täckström, Das, & Uszkoreit, 2016).

Previous work on text representation can be categorized into three main types (Xie et al., 2016), i.e., *statistics-based, neural-network-based* and *pretraining-based* embeddings. First, statistics-based embedding models are estimated based on a statistical indicator, e.g., the frequency of co-occurring words (in bag-of-words models (Joachims, 1998)), the frequency of co-occurring word pairs (in *n*-gram models (Zhang, Zhao, & Lecun, 2015)), and the weights of words in different documents (the TF-IDF

[*] Corresponding author.

E-mail addresses: zhengjianming12@nudt.edu.cn (J. Zheng), caifei@nudt.edu.cn (F. Cai), chenhonghui@nudt.edu.cn (H. Chen), derijke@uva.nl (M. de Rijke).

https://doi.org/10.1016/j.ipm.2020.102215
Received 12 October 2019; Received in revised form 3 January 2020; Accepted 26 January 2020

model (Robertson, 2004)). Second, neural-network-based embedding models mainly rely on a neural network architecture to learn a text representation, which is either based on a hidden layer (Joulin, Grave, Bojanowski, & Mikolov, 2017), convolutional neural networks (CNNs) (Kim, 2014) or recurrent neural networks (RNNs) (Liu, Qiu, & Huang, 2016). Additionally, this type of methods also consider the syntactic structure to reflect the semantics of text, e.g., recursive neural networks (Socher et al., 2013) and tree-structured long short-term memory networks (Tree-LSTM) (Tai, Socher, & Manning, 2015). Pretraining-based embedding models adopt a feature-based (McCann, Bradbury, Xiong, & Socher, 2017; Mikolov, Chen, Corrado, & Dean, 2013; Pennington, Socher, & Manning, 2014; Peters et al., 2018) or fine-tuning strategy (Dai & Le, 2015; Devlin, Chang, Lee, & Toutanova, 2019; Howard & Ruder, 2018; Yang et al., 2019) to capture the semantics and syntactic information from a large text corpora.

In general, the aforementioned models work well for the task of text classification. (Howard & Ruder, 2018; Joulin et al., 2017; Kim, 2014; Zhang et al., 2015) However, in existing embedding models, the generated process of the vectorized representation of a text usually follows a so-called one-way action. That is to say, representations generated from the preceding text are taken as the context to determine the representations of later texts. Although a bidirectional LSTM considers bidirectional actions, it simply concatenates two one-way actions to get the embeddings, e.g., (Liu et al., 2016). We argue that the semantics as defined in terms of a text representation should be a product of interactions of all source elements (e.g., words or sentences) in the text. Typically, restriction to one-way actions may result in a partial semantic loss (Saif, He, Fernández, & Alani, 2016), causing the poor performances in the downstream applications. We hypothesize that although these interaction relations may be learned by neural networks with enough samples, modeling explicitly such interaction relations can directly make text representation more informative and effective. Furthermore, recent unsupervised representation learning has proven to be effective and promising in the field of natural language processing (Devlin et al., 2019; Howard & Ruder, 2018; McCann et al., 2017; Peters et al., 2018; Yang et al., 2019). However, these approaches are limited to a single strategy (either feature-based or fine-tuning strategy), which results in a so-called fine-tune error, which may be trapped in the local best.

Thus, as illustrated in Fig. 1, we focus on the task of text classification and propose a novel pipeline with the following ingredients:

1. **pre-train** language model on a large text corpus to get the related word embeddings and neural networks parameters;
2. **interact** the word embeddings based on the pre-trained parameters to obtain the interaction representation; and
3. **fine-tune** the classifier with the interaction representation and pre-trained word embeddings as input.

More specifically, in the interaction representation layer, we propose a two-perspective interaction representation using a *Local Interaction Representation* (LIR) and a *Global Interaction Representation* (GIR). First, the LIR applies an attention mechanism (Bahdanau, Cho, & Bengio, 2015) inside the syntactic structure of a sentence, e.g., the dependency-based parse trees or constituency-based parse trees, to reflect the local interaction of adjacent words. Second, the GIR employs an attention mechanism with an enumeration-based strategy to represent the interactions of all words in a sentence. After that, we make an initiative combine LIR and GIR to into a *Hybrid Interaction Representation* (HIR) model to represent both local and global interactions of words in a sentence. For the pretrain-finetuning process, we combine the feature-based and the fine-tuning strategies and propose a *hybrid language model pretrain-finetuning* (HLMPf) approach. HLMPf first follows the fine-tuning strategy to employ the pre-trained embeddings and neural network parameters as the initialization of the interaction representation layer. Then, according to the feature-based strategy, HLMPf applies the pre-trained embeddings as additional features and concatenates the interaction representation in the classifier fine-tuning layer.

For evaluation, we conduct a comprehensive experiment on five publicly available benchmark datasets for the task of text classification. The experimental results show that our proposals with interaction representations and the hybrid pretrain-finetuning strategy outperform the state-of-the-art baselines for text classification, with improvements ranging from 2.03% to 3.15% in terms of error rate.

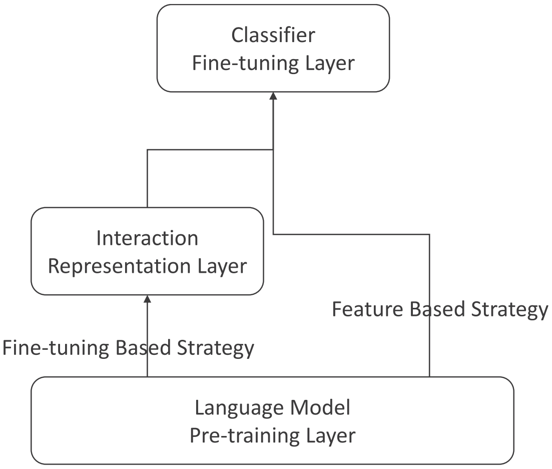


Fig. 1. Overview pipeline of pre-train interact fine-tune.

The main contributions of our work can be summarized as follows:

1. We propose a novel pipeline for the task of text classification, i.e., *Pre-train, Interact, Fine-tune* (PIF).
2. To the best of our knowledge, ours is the first attempt to model word interactions for text representation. We introduce a two-perspective interaction representation for text classification, i.e., a Local Interaction Representation (LIR) and a Global Interaction Representation (GIR), which are then combined to generate a Hybrid Interaction Representation (HIR) model.
3. We combine the meritsof two popular language model pretrain-fine-tuning strategies (feature-based and fine-tuning) and propose the hybrid language model pretrain-finetuning (HLMPf).
4. We analyze the effectiveness of our proposals and find that they achieve better results compared with other state of art text classification methods in terms of error rate.

2. Related work

In this section, we briefly summarize the general statistical approaches for text representation in Section 2.1 followed by the neural-networks-based methods in Section 2.2. We then describe the recent work on language model pre-training for downstream applications in Section 2.3.

2.1. *Statistics-based representation*

As a word is the most basic unit of semantics, the traditional one-hot representation model converts a word in a vocabulary into a sparse vector with a single high value (i.e., 1) in its position and the others with a low value (i.e., 0). The representation is employed in the Bag-of-Words (BoW) model (Joachims, 1998) to reflect the word frequency. However, the BoW model only symbolizes the word hence incapable of the semantic relationship between words. Consequently, the bag-of-means model (Zhang et al., 2015) was proposed to cluster the word embeddings learned by the word2vec model (Mikolov et al., 2013). Furthermore, the bag-of-n-grams (Zhang et al., 2015) was developed to take the *n*-grams (up to 5-grams) as the vocabulary in the BoW model. In addition, with some extra statistical information, e.g., TF-IDF, a better document representation can be produced (Robertson, 2004). Other text features, e.g., the noun phrases (Lewis, 1992) and the tree kernels (Post & Bergsma, 2013), were incorporated into the model construction.

Clearly, a progressive step has been made in statistical based representation (Bernauer, Han, & Sohn, 2018). However, such traditional statistical approaches, on one hand, inevitably face the problems of data sparsity and dimensionality, leading to no applications on large-scale corpora. On the other hand, such approaches are simply built on shallow statistics, and a deeper semantic information of the text has not been well developed.

Instead, this paper attempts to provide solutions based on the fact that neural networks possesses the ability to learn a low-dimensional and distributed representation to overcome such problems.

2.2. *Neural-based representation*

Since Bengio, Ducharme, and Vincent (2000) first adopted the neural network architecture to train a language model, considerable attention has been devoted to proposing neural network-related models for text representation. For instance, the FastText model (Joulin et al., 2017) employs one hidden layer to integrate the subword information and obtains satisfactory results. However, this model simply averages all word embeddings and discards the word order. In view of that, Liu et al. (2016) employed the recurrent structure, i.e., RNNs, to consider the word order and to jointly learn text representation across multiple related tasks. Compared to RNNs, CNNs are easier to train and capture the local word-pair information (Kim, 2014; Zhang et al., 2015).

Furthermore, a combination of neural network models are integrated to develop the advantage of each single neural network. For example, Lai, Xu, Liu, and Jun (2015) proposed the recurrent convolutional neural networks (RCNN), which adopted the recurrent structure to grasp the context information and employed a max-pooling layer to identify the key components in text. Besides, other document features have been injected into the document modeling. For instance, Zheng, Cai, Chen, Feng, and Chen (2019) took the hierarchical structure of text into account. He, Lee, Ng, and Dahlmeier (2018) and Saraiva, Cavalcanti, de Moura, Gonçalves, and da Silva Torres (2016) transformed the document-level knowledge to improve the performance of aspect-level sentiment classification.

Although these approaches have been proven to be effective in the downstream applications, they completely depend on the structure of network to implicitly represent a document, ignoring the interaction that exists among the source elements in a document, e.g., words or sentences. To tackle these problem, our proposals are capable of modeling the interaction as the starting point to better reflect the semantic relationship between words in a sentence, which we argue is of great significance in helping to improve the performance of downstream tasks, e.g, sentimental classification.

2.3. *Language model pre-training-based representation*

The language pre-training model has been shown effective in the natural language processing tasks, e.g., question answering (McCann et al., 2017), textual entailment (Peters et al., 2018), semantic role labeling (Devlin et al., 2019) sentimental analysis (Dai & Le, 2015), etc. These pre-training models can be mainly classified into two classes, i.e., feature-based models and fine-tuning models.

The feature-based models generate the pre-trained embeddings from other tasks, where the output can be regarded as the additional features for the current task architecture. For instance, word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) focus on transforming words into the distributed representations and capturing the syntactics as well as the semantics by pre-training the neural language models on a large text corpora. In addition, McCann et al. (2017) concentrated on the machine translation task to get the contextualize word vectors (CoVe). Since these word-level models suffer from the word-ploysemy issues, Peters et al. (2018) developed the sequence-level model, i.e., ELMo, to capture the complex word features across different linguistic contexts and then use ELMo to generate the context-sensitive word embeddings.

Different from the feature-based strategy (Mehta & Majumder, 2018), the fine-tuning models first produce the contextual word presentations which have been pre-trained from unlabeled text and fine-tune for a supervised downstream task. For instance, Dai and Le (2015) trained a sequence auto-encoder model on unlabeled text as an initialization of another supervised network. However, this method suffers from overfitting and requires some in-domain knowledge to improve the performance. Consequently, Universal Language Model Fine-tuning (ULMFit) (Howard & Ruder, 2018) was developed, which leveraged the general-domain pre-training and the novel fine-tuning techniques to tackle the overfitting issue. In addition, Devlin et al. (2019) proposed two unsupervised tasks, i.e., masked language model and next sentence prediction, to further improve the fine-tuning process. Besides, XLNet (Yang et al., 2019) was proposed to employ the permutation language model to capture the bidirectional context and avoid the pretrain-finetune discrepancy.

While the language pre-training model based representations have been proposed and proven promising in the NLP tasks, these methods are limited to either feature-based or fine-tune-based strategy. Ultimately, we can nonetheless make steps further by incorporating their respective characteristics to improve the performance of downstream applications.

3. Proposed models

In this section, we first formally describe the processto compute the interaction representation in Section 3.1, which can be divided into three parts, i.e., LIR (see Section 3.1.1),GIR (see Section 3.1.2) and HIR (see Section 3.1.3). Then, we introduce the HLMPf approach in detail (see Section 3.2), which is the combination of the feature-based and fine-tuning strategies.

For clarity, we also summarize some notations mentioned in this section on the Table 1.

3.1. Interaction representation

We describe the Local Interaction Representation (LIR) of adjacent words and introduce the Global Interaction Representation (GIR) of all words in a sentence. After that, a Hybrid Interaction Representation (HIR) model is proposed.

3.1.1. Local interaction representation

We introduce an attentive tree LSTM that computes a local representation of words. The idea of an *action* of a word on another word is that the former assigns a semantic weight to the latter.

The experiments we conduct related to LIR are centered on constituency-based trees, but we explain the core concepts for both dependency-based and constituency-based trees. Given a dependency-based parse tree, let $C(p)$ denote the set of child words of a parent word x_p. To define the attentive tree LSTM, we introduce hidden states and memory cells h_k and c_k ($k \in \{1, 2,..., |C(p)|\}$) for every child word, respectively. As shown in Fig. 2, unlike the Tree-LSTM model in Tai et al. (2015) that only performs the one-way action (child words ↦ parent word), LIR also considers an action in the opposite direction, i.e., parent word ↦ child words.

Let us explain this in detail. In an action parent word ↦ child words, we regard the parent word x_p as a controller that assigns semantic weights based on the attention mechanism to its child words in a sentence. Thus, we first convert the parent word x_p into a hidden representation $\overline{h_p}$ as follows:

$$\overline{h_p} = \tanh(W^{(h)}P_{x_p} + b^{(h)}), \tag{1}$$

where P_{x_p} is the pre-trained word embedding for parent word x_p; $W^{(h)}$ and $b^{(h)}$ are the weight matrix and the bias term, respectively. Then, we employ a general content-based function (Luong, Pham, & Manning, 2015) to connect the parent word and the child words

Table 1
Variable descriptions.

Variable	Description
$\{x_1, x_2, \cdots, x_n\}$	the word sequence in sentence s
P_{x_i}	the pre-trained embeddings for word x_i
I_{x_i}	the interaction representation for word x_i
C_{x_i}	the context-aware embeddings for word x_i
W_{x_i}	the trainable word embeddings for word x_i
$C(p)$	the set of child words of a parent word x_p
ψ	The trainable parameters ψ for all neural-network layers, including all kinds of trainable matrices and bias

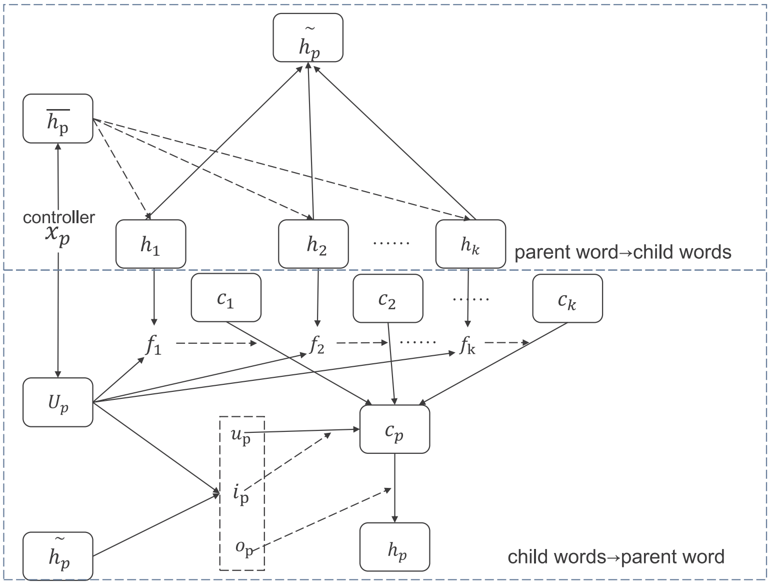


Fig. 2. Structure of the local interaction representation model. (For simplicity, we write U_p for $(U^{(i)}P_{x_p}, U^{(o)}P_{x_p}, U^{(u)}P_{x_p}, U^{(f)}x_p)$.).

as follows:

$$\alpha_k = \overline{h_p} W_\alpha h_k, \tag{2}$$

where α_k is the connective representation of $\overline{h_p}$ and the hidden state h_k, and W_α is the connective matrix to be learned. After that, we apply a softmax function on a sequence of connective representations $\{\alpha_1, \alpha_2, \ldots, \alpha_{|C(p)|}\}$ to get the weight λ_k as follows:

$$\lambda_k = \frac{\exp(\alpha_k)}{\sum_{i=1}^{|C(p)|} \exp(\alpha_i)}. \tag{3}$$

Finally, we represent the hidden interaction state $\tilde{h}_p$ that relates to all child states of the parent word x_p, i.e.,

$$\tilde{h}_p = \sum_{i \in C(j)} \lambda_i h_i. \tag{4}$$

In the action child words ↦ parent word in Fig. 2, we utilize the hidden interaction state $\tilde{h}_p$ and the parent word x_p as input to the LSTM cell and obtain

$$i_p = \sigma(U^{(i)} x_p + W^{(i)} \tilde{h}_p + b^{(i)}), \tag{5}$$

$$o_p = \sigma(U^{(o)} x_p + W^{(o)} \tilde{h}_p + b^{(o)}), \tag{6}$$

$$u_p = \tanh(U^{(u)} x_p + W^{(u)} \tilde{h}_p + b^{(u)}), \tag{7}$$

$$f_{kp} = \sigma(U^{(f)} x_p + W^{(f)} h_k + b^{(f)}), \tag{8}$$

where i_p, o_p and f_{kp} are the input gate, the output gate and the forget gate, respectively; u_p is the candidate hidden state of x_p. For i_p, o_p, u_p and f_{kp}, we have a corresponding weight matrix of x_p (i.e., $U^{(i)}$, $U^{(o)}$, $U^{(u)}$ and $U^{(f)}$), a weight matrix of $\tilde{h}_p$ (or h_k) (i.e., $W^{(i)}$, $W^{(o)}$, $W^{(u)}$ and $W^{(f)}$), and a bias term (i.e., $b^{(i)}$, $b^{(o)}$, $b^{(u)}$ and $b^{(f)}$). Finally, we can get the memory cell c_p and the hidden state h_p of the parent word x_p as follows:

$$c_p = i_p \odot u_p + \sum_{k=1}^{|C(p)|} f_{kp} \odot c_k, \tag{9}$$

$$h_p = o_p \odot \tanh(c_p), \tag{10}$$

where $\odot$ is element-wise multiplication and c_k is the memory cell of a child word.

Similarly, given a constituency-based tree, let x_l and x_r denote the left child word and the right child word of a parent word x_p. Since the parent word x_p is a non-terminal node (i.e., x_p is a zero vector), we use x_l and x_r as the controller instead of P_{x_p}, respectively. Therefore, following Eqs. (2)–(4), we obtain the hidden interaction states $\tilde{h}_l$ and $\tilde{h}_r$ related to x_l and x_r, respectively. We concatenate $\tilde{h}_l$ and $\tilde{h}_r$ to represent the hidden interaction states of the parent word, i.e., $\tilde{h}_p = [\tilde{h}_l; \tilde{h}_r]$. Again, following Eqs. (5)–(10), we can get the memory cell c_p and the hidden state h_p for parent word x_p.

At this stage we have represented the local interaction, and each word has been updated by the interaction representation.

3.1.2. Global interaction representation

Different from LIR, which captures the syntactic relation between words, GIR adopts a enumeration-based strategy to employ an attention mechanism on all words in a sentence.

In detail, after implementing Tree-LSTM on all n words in a sentence, we are able to obtain the hidden representations $\{h_1, h_2, \ldots, h_n\}$ corresponding to the words $\{x_1, x_2, \ldots, x_n\}$. In order to represent the interaction between a word x_g and the other words in a sentence, we regard the word x_g as a controller that can assign semantic weights to other words in $\{x_1, x_2, \ldots, x_n\}$ excluding x_g itself. Similarly, we employ a general content-based function to connect the word x_g with other words as follows:

$$\alpha_{gk} = h_g W_\alpha h_k, \tag{11}$$

where α_{gk} is the connective representation of h_g and h_k $(g, k \in (1, 2, \ldots, n))$. After that, we can get all connective representations $\{\alpha_{g1}, \alpha_{g2}, \ldots, \alpha_{gn}\}$ between the word x_g and other words. Then, we can apply a softmax function on the connective representation sequence to calculate the weight as follows:

$$\lambda_{gk} = \frac{\exp(\alpha_{gk})}{\sum_{i=1}^{n} \exp(\alpha_{gi})}, \tag{12}$$

where λ_{gk} is the weight of word x_k in $\{x_1, x_2, \ldots, x_n\}$ that interacts with word x_g. Finally, we obtain the interaction representation r_g as follows:

$$r_g = \sum_{i=1}^{n} \lambda_{gi} h_i. \tag{13}$$

By doing so, we enumerate all words in a sentence and can return a sequence of interaction representations as $\{r_1, r_2, \ldots, r_n\}$. We then adopt a max-pooling (Ranzato, Boureau, & Lecun, 2007) on this sequence to produce the sentence embeddings s by

$$s = \max\{r_1, r_2, \ldots, r_n\}. \tag{14}$$

This completes our definition of the global interaction representation. We can train the sentence representation s to update the pre-trained embeddings.

3.1.3. Hybrid interaction representation

In order to capture both local and global interactions between words, we combine LIR and GIR to form a hybrid interaction representation model (HIR) for text representation. HIR first follows the procedure of LIR to produce the hidden state representations $\{h_1, h_2, \ldots, h_n\}$ for the corresponding word $\{x_1, x_2, \ldots, x_n\}$. Then, HIR employs the process of GIR on these hidden state representations to get the final sentence embeddings s.

Eventually, in the process of class prediction, we apply a softmax classifier on the sentence embeddings s to get a predicted label $\hat{s}$, where $\hat{s} \in \mathcal{Y}$ and $\mathcal{Y}$ is the class label set, i.e.,

$$\hat{s} = \arg\max p(\mathcal{Y}|s), \tag{15}$$

where

$$p(\mathcal{Y}|s) = \mathrm{softmax}(W^{(s)}s + b^{(s)}). \tag{16}$$

Here, $W^{(s)}$ and $b^{(s)}$ are the reshape matrix and the bias term, respectively. For formulating the loss function in HIR, we combine the corresponding loss in LIR and GIR as

$$L = \frac{\gamma}{n} \sum_{i=1}^{n} \log p(\tilde{w}_i|h_i) + (1-\gamma)\log p(\tilde{s}|s), \tag{17}$$

where the former loss $(\frac{1}{n}\sum_{i=1}^{n} \log p(\tilde{w}_i|h_i))$ comes from LIR and the latter loss $(\log p(\tilde{s}|s))$ from GIR, γ is the trade-off parameter between LIR and GIR. In LIR, h_i is the hidden state and $\tilde{w}_i$ is the true class label of word x_i. While in GIR, $\tilde{s}$ is the true class label of sentence embeddings s. In addition, both $\tilde{w}_i$ and $\tilde{s}$ can be trained using the dataset.

We have now introduced the main process of our HIR model. Clearly, as shown in Algorithm 1, we first employed Bi-LSTM process the pre-trained word sequence to build their semantics relations from step 1 to 2. Then, with the help of syntactic parse tool, we are able to obtain the parent-child set T. Following the bottom-up traversal algorithm, we show how to model the local interaction representation between parent word and child words from step 4 to 11. While from step 12 to 19, we show how to compute the global interaction representation between all words. At last, we optimize the loss function to jointly training the process of LIR and GIR. And update the pre-trained embeddings with the interaction representations.

3.2. Hybrid language model pretrain-finetuning

Unsupervised representation learning, as a fundamental tool, has been shown effective in many language processing tasks (Devlin et al., 2019; Howard & Ruder, 2018; McCann et al., 2017; Peters et al., 2018; Yang et al., 2019). Here, we propose the *hybrid language*

```
Input: The pre-trained embeddings for each word in a sentence $s_t$, i.e., $\{P_{x_1}, P_{x_2}, \ldots, P_{x_n}\}$; the pre-trained parameters $\psi$ of the
    neural networks from the language pre-training layer.
Output: The interaction representation for the word sequence, i.e., $\{I_{x_1}, I_{x_2}, \ldots, I_{x_n}\}$
$\overrightarrow{h_t} \leftarrow \overrightarrow{LSTM}(x_t, h_{t-1})$, $\overleftarrow{h_t} \leftarrow \overleftarrow{LSTM}(x_t, h_{t-1})$
$h_t = (\overrightarrow{h_t}; \overleftarrow{h_t}), t = 1, 2, \ldots, n$
Get the set of parent word $p_i$ and its child words by syntactic parsing:
$T = \{p_i, C(p_i)\} \leftarrow$ syntactic parse $(s)$, $i = 1, \ldots, |T|$
for each parent word $\{p_i, C(p_i)\} \in T$ do
    $\overline{h_{p_i}} = \tanh(W^{(h)} P_{x_{p_i}} + b^{(h)})$
    for each child word in $C(p_i)$: do
        Get the connective representation: $\alpha_k = \overline{h_p} W_\alpha h_k$
    end for
    $\tilde{h}_p = \sum_{i \in C(j)} \lambda_i h_i$, where $\lambda_k = \frac{\exp(\alpha_k)}{\sum_{i=1}^{|C(p)|} \exp(\alpha_i)}$
                              %% parent word ↦ child words
    $h_p \leftarrow LSTM(x_p, \tilde{h_p})$ %% child words ↦ parent word
end for
%% This loop for LIR that follows the bottom-up algorithm to traverse the syntactic parsing tree.
for each word $w_g$ in sentence $s_t$ do
    Word $w_g$ is regarded as the parent word
    for each word in sentence $s_t$ excluding word $w_g$ do
        $\alpha_{gk} = h_g W_\alpha h_k$
    end for
    $r_g = \sum_{i=1}^{n} \lambda_{gi} h_i$, where $\lambda_{gk} = \frac{\exp(\alpha_{gk})}{\sum_{i=1}^{n} \exp(\alpha_{gi})}$
    $s = \max\{r_1, r_2, \ldots, r_n\}$
end for   %% This loop for GIR.
Optimize the loss function: $L = \frac{\gamma}{n} \sum_{i=1}^{n} \log\ p(\tilde{w}_i | h_i) - (1 - \gamma) \log\ p(\tilde{s} | s)$
Update $I_{x_i} \leftarrow h_{x_i}$
return $\{I_{x_1}, I_{x_2}, \ldots, I_{x_n}\}$
```

Algorithm 1. Hybrid Interaction Representation.

```
Input: The text need to be trained.
Output: The trained parameters $\psi$ for all neural-network layers; the trained word embeddings $\{W_{x_1}, W_{x_2}, \ldots, W_{x_n}\}$.
Pre-train the masked language model and next sentence prediction tasks to get the pre-trained neural networks and related parameters.
Add the interaction representation layer to the pre-training layer.
Following algorithm 1, optimize the loss function to fine-tune the related parameters.
                %% the fine-tuning strategy
Pre-train some supervised tasks to get the context-aware word embeddings, i.e., $\{C_{x_1}, C_{x_2}, \ldots, C_{x_n}\}$ (similar to ELMo (Peters et al., 2018)).
Add the classifier fine-tuning layer to the former combination layer.
Use the $W_{x_i} = \{I_{x_i}; C_{x_i}\}$ as the input of the classifier fine-tuning layer to further fine-tune $\psi$ and $\{W_{x_1}, W_{x_2}, \ldots, W_{x_n}\}$.
                %% the feature-based strategy
return $\psi$ and $\{W_{x_1}, W_{x_2}, \ldots, W_{x_n}\}$
```

Algorithm 2. Hybrid Language Model Pretrain-finetuning.

Table 2
An overview of models discussed in the paper.

Model	Description	Source	Pretrain-finetuning
Bi-LSTM	A Bi-LSTM based representation model.	(Liu et al., 2016)	×
C-CNN	A CNN based representation model in the character level.	(Zhang et al., 2015)	×
CoVe	A text representation model transferred from the machine translation model.	(McCann et al., 2017)	√
ULMFiT	A text representation model based on general-domain language model pre-train, target task language model and classifier fine-tune.	(Howard & Ruder, 2018)	√
LIR	A text representation model based on the local interaction representation.	This paper	×
GIR	A text representation model based on the global interaction representation.	This paper	×
HIR	A text representation model based on the hybrid interaction representation.	This paper	×
LIR_B	A text representation model based on the local interaction representation model in the BERT fine-tuning architecture.	This paper	√
GIR_B	A text representation model based on the global interaction representation model in the BERT fine-tuning architecture.	This paper	√
HIR_B	A text representation model based on the hybrid interaction representation model in the BERT fine-tuning architecture.	This paper	√
LIR_P	A text representation model based on the local interaction representation model in the Pre-train Interact Fine-tune architecture.	This paper	√
GIR_P	A text representation model based on the global interaction representation model in the Pre-train Interact Fine-tune architecture.	This paper	√
HIR_P	A text representation model based on the hybrid interaction representation model in the Pre-train Interact Fine-tune architecture.	This paper	√

model pretrain-finetuning (HLMPf) method, which integrates the respective advantages in the PIF pipeline shown in Fig. 1. The details of HLMPf are shown in Algorithm 2.

We first follow the BERT (Devlin et al., 2019) model to train the language model pre-training layer. From step 2 to 3, we employ the fine-tuning strategy to fine-tune the interaction representation layer and the language model pre-training layer. After that, we follow the ELMo approach (Peters et al., 2018) to obtain the context-aware word embeddings. From step 5 to 6, following the feature-based strategy, we show how to further fine-tune all neural layers.

Specially, since fine-tuning all layers at once will result in catastrophic forgetting, we adopt the gradual unfreezing strategy (Howard & Ruder, 2018) to fine-tune all neural layers.

4. Experiments

We start off our experiments by providing an overview of the text representation model to be discussed in this paper and listing some the research questions that guide our experiments. Then we describe the task and datasets that we evaluate our proposals on. We conclude this section by specifyingthe settings of the parameters in our experiments.

4.1. Model summary and research questions

Table 2 list the models to be discussed. Among these models, Bi-LSTM, Char-level CNN, LIR, GIR and HIR models are neural based representation and don’t experience the pretrain-finetuning process.

Baselines Four state-of-the-art baselines: two neural based representation model (i.e., Bi-LSTM (Liu et al., 2016), C-CNN (Zhang et al., 2015)), two language model pre-training based representation model (i.e., CoVe (McCann et al., 2017), ULMFiT (Howard & Ruder, 2018)).
Our proposals Nine flavors of approaches that we introduce in this paper: three interaction representation models (i.e., LIR, GIR and HIR), three interaction representation models in the BERT architecture (i.e., LIR_B, GIR_B, HIR_B) and the Pre-train, Interact, Fine-tune (PIF) architecture (i.e., LIR_P, GIR_P, HIR_P).

To provide a qualitative analysis of our proposed interaction representation models and the PIF architecture, we consider a text classification task and seek to answer the following questions:

RQ1 Does the interaction representation incorporated in the text representation model help to improve the performance for text classification?
RQ2 Compared with the existing pretrain-finetuning approaches, does our proposed PIF architecture help to improve the model performance for text classification?
RQ3 How does the trade-off parameter between LIR and GIR (as encoded in γ) impact the performance of HIR related model in terms of classification error rate?
RQ4 Are the performance of our proposals sensitive to the length $L$of text to be classified?

4.2. Datasets

We evaluate the effectiveness of our proposals with five public available datasets used in different application domains, e.g., sentiment analysis, questions classification and topic classification, which are widely used by the state-of-the-art models for text classification, e.g., CoVe (McCann et al., 2017) and ULMFiT (Howard & Ruder, 2018). Table 3 details the statistics of the datasets. We use error rate as the evaluation metric to compare the performance of discussed models.

Sentiment analysis Sentiment analysis mainly concentrates on the movie review and shopping review datasets. For example, IMDb dataset proposed by Maas et al. (2011) is a movie review dataset with binary sentimental labels. While Yelp dataset compiled by Zhang et al. (2015) is a shopping review dataset that has two versions, i.e., binary and five-class version. We concentrate on the five-class version (Johnson & Zhang, 2017).
Question classification For question classification, Voorhees and Tice (1999) collected open-domain fact-based questions and divided them into broad semantic categories, which has six-class and fifty-class versions. We mainly focus on the small six-class version and hold out 452 examples for validation and leave 5000 for training, which is similar to McCann et al. (2017).
Topic classification For topic classification, we evaluate our proposals on the task of news article and ontology classification. We use the AG news corpus collected by Zhang et al. (2015), which has four classes of news with only the titles and description fields. In addition, the DBpedia dataset, collected by Zhang et al. (2015), is used, which contains the title and abstract of each Wikipedia article with 14 non-overlapping ontology classes. In general, the dataset division is the same as in Zhang et al. (2015).

4.3. Model configuration and training

For data preprocessing, we split the text into sentences and hence tokenized each sentence using Stanford's CoreNLP (Manning et al., 2014). In addition, we discard the words with single characters and other punctuation and convert the upper-case letters to the lower-cases letters. In order to fit in the BERT pre-training, we add a special token for each sentence, e.g., [CLS] and [SEP]. The other data preprocessing follow the same way as Johnson and Zhang (2017)

For model configuration, we use the same set of hyper-parameters across all datasets to evaluate the robustness of our proposals. In the process of pre-training, we directly employ the trained $BERT_{base}$[1] as our language model pre-training layer for simplicity. As for the feature-based process, we follow the ELMo model[2] and employ AWD-LSTM (Merity, Keskar, & Socher, 2018) on the trained BERT layer to get the context-aware word embeddings. For classifier fine-tuning layer, we adopt a softmax classifier and set the size of hidden layer to 100. In addition, we set the dimension of word embeddings and hidden representation in the interaction representation layer to 400 and 200, respectively. We also apply a dropout of 0.4 to layers and 0.05 to the embedding layers.

For the whole training process, we use a batch size of 64, a base learning rate of 0.004 and 0.01 for fine-tuning the interaction representation layer and the classifier fine-tuning layer, respectively. We employ a batch normalization mechanism (Ioffe & Szegedy, 2015) to accelerate the training of the neural networks. Gradient clipping is applied by scaling gradients when the norm may exceed a threshold of 5 (Pascanu, Mikolov, & Bengio, 2013). For the fine-tuning process, we adopt the gradual unfreezing strategy (Howard & Ruder, 2018) to fine-tune all neural layers.

5. Results and discussion

In Section 5.1, we examine the performance of our proposals incorporated with the interaction representation and the HLMPf on five public datasets, which aim at answering **RQ1** and **RQ2**. Then, in Section 5.2, we analyze the impact of the trade-off parameter γ in HIR related model to answer **RQ3**. Finally, to answer **RQ4**, section Section 5.3 focuses on investigating the impact on the text classification by varying the text length.

5.1. Performance comparison

5.1.1. Performances about the interaction representation

To answer **RQ1**, we first compare the performance of the basic interaction representation based models (i.e., LIR, GIR and HIR) with the baselines and present the results in Table 4

As to the baselines, we present two types of representation models, i.e., the neural-network based model (Bi-LSTM and C-CNN) and the pretrain-finetuning based model (CoVe and ULMFiT). For the neural-network based model, C-CNN achieves better results as compared to Bi-LSTM. While in the pretrain-finetuning based model, ULMFiT is obviously the better one. Interestingly, comparing these two types of models, we can find that the representation models with pretrain-finetuning process present obvious advantages in terms of reducing error rate. Specially, with regard to C-CNN, ULMFiT reduces the error rate dramatically by 37.5%, 26.6%, 44.4%, 89.8% and 48.4% on the corresponding datasets (IMDb, Yelp, TREC, AG and DBpedia in order, which is the same in the following text). This may be due to the fact that the pre-training on a large text corpora is capable of capture the deep syntactic and semantic information, which cannot be realized by only training on the neural networks.

[1] https://github.com/google-research/bert.
[2] https://github.com/allenai/bilm-tf.

Table 3
Dataset statistics.

Dataset	IMDb	Yelp	TREC	AG	DBpedia
Type	Sentiment	Sentiment	Question	Topic	Topic
# training documents	25K	560K	5K	120K	560K
# text documents	2K	50K	0.5K	7.6K	70K
# classes	2K	5K	6.5K	4K	14K

Table 4
Error rate (%) about the interaction representation on different datasets. (The results of the best baseline and the best performer in each column are underlined and boldfaced, respectively. Results marked with * are re-printed from Zhang et al. (2015), McCann et al. (2017), Howard and Ruder (2018) and Zhou et al. (2016). The rest are obtained by our own implementation. Statistical significance of pairwise differences (the best proposed model vs. the best neural-network-based baseline) are determined by a *t*-test (▲/▼ for $\alpha = 0.01$).

Datasets	IMDb	Yelp	TREC	AG	DBpedia
Bi-LSTM	8.72	41.83*	7.66	13.94*	1.45*
C-CNN	7.36	37.95*	6.48	9.51*	1.55*
CoVe	8.2*	–	4.2*	–	–
ULMFiT	***4.6****	***29.98****	***3.6****	***5.01****	***0.80****
LIR	6.86	35.58	5.76	7.83	1.31
GIR	6.92	35.46	5.87	8.20	1.37
HIR	6.73▲	34.18▲	5.44▲	7.53▲	1.24▲

Similarly, our proposals with restricted interaction representation, i.e., LIR, GIR and HIR, limits its ability to compete with the state-of-the-art pretrain-finetuning based model, i.e., ULMFiT. But for the neural-network based baselines, our proposals demonstrate better performance in terms of error rate. In particular, HIR is the best performing model among our proposals, which shows an improvement against the best neural-network based baseline, i.e., C-CNN, resulting in 8.6%, 9.9%, 16.0%, 26.3% and 20% reduction in terms of error rate on the respective datasets. LIR and GIR, following HIR, can outperform C-CNN on all datasets. The aforementioned findingsin contrast with the traditional neural-network based models, modeling the interaction process explicitly better captures the semantics relation between source elements in the text and generate more meaningful text representation. HIR, in particular,by representing the local and global interaction between words, it is more effective to improve the performance of the downstream applications.

5.1.2. Performances about the pretrain-finetuning

In Section 5.1.1, the effectiveness of the pretrain-finetuning based and the interaction-related models have been proven. However, the basic interaction representation based models cannot beat the state-of-the-art pretrain-finetuning based model, i.e., ULMFiT. Hence, we incorporate them with the popular pretrai-finetuning architecture (i.e., BERT) and our PIF architecture to get the corresponding models (i.e., LIR$_B$, GIR$_B$, HIR$_B$ and LIR$_P$, GIR$_P$, HIR$_P$), respectively. To answer **RQ2**, we compare the performance of these proposed models with ULMFiT and present their experimental results in Table 5.

Clearly, as seen in Table 5, it turns out that our basic interaction-related models incorporated with the pretrain-finetuning process generally outperform the state-of-the-art model, i.e., ULMFiT, except for some cases, e.g., the LIR$_B$ on DBpedia, GIR$_B$ on AG and DBpedia, GIR$_P$ on DBpedia. This findings confirms the outlook of our basic interaction representation models is promising under the pretrain-finetuning architecture. With regard to the BERT architecture, our interaction-related models present the similar error rate distribution to the basic interaction representation models in Table 4. HIR$_B$ is the best performer using the BERT architecture, followed by LIR$_B$ and GIR$_B$. Specially, for each dataset, HIR$_B$ shows an obvious improvement of 7.9%, 5.6%, 6.4%, 2.6% and 2.5% against ULMFiT, respectively. While LIR$_B$, except on the DBpedia, also gains a minor improvement of 0.4%, 5.2%, 1.7%, 1.6% against

Table 5
Error rate (%) about the pretrain-finetuning process on different datasets. (The results of the best baseline and the best performer in each column are underlined and boldfaced, respectively. Results marked with * are re-printed from Howard and Ruder (2018). The rest are obtained by our own implementation. Statistical significance of pairwise differences (the best proposed model vs. the best neural-network-based baseline) are determined by a *t*-test (▲/▼ for $\alpha = 0.01$).

Datasets	IMDb	Yelp	TREC	AG	DBpedia
ULMFiT	4.6*	29.98*	3.6*	5.01*	0.80*
LIR$_B$	4.58	28.42	3.54	4.93	0.81
GIR$_B$	4.69	28.84	3.55	5.03	0.84
HIR$_B$	4.24	28.31	3.37	4.88	0.78
LIR$_P$	4.25	27.33	3.40	4.92	0.80
GIR$_P$	4.31	27.66	3.48	4.94	0.81
HIR$_P$	**4.04**▲	**27.07**▲	**3.33**▲	**4.85**▲	**0.77**▲

ULMFiT, respectively. GIR$_B$, a bit worser than LIR$_B$, beats the ULMFiT on 3 out of 5 datasets.

The similar findings can also be found in the PIF architecture. In particular, HIR$_P$ achieves the best performance not only in the PIF architecture but among all discussed models. Compared with the baseline ULMFiT, HIR$_P$ gains substantial improvements of 12.1%, 9.7%, 7.5%, 3.2%, 3.8% in terms of error rate on respective datasets. In addition, LIR$_P$ wins the comparisons against ULMFiT, resulting in 7.6%, 8.8%, 5.6%, 1.8% improvements on the respective datasets and an equal performance on DBpedia. While GIR$_P$ defeats the ULMFiT model on 4 out 5 datasets.

Furthermore, comparing the same type of interaction models with different architectures (e.g., type LIR: LIR, LIR$_B$, LIR$_P$), we can find that there exists a unchanged ranking order of performance on each dataset, i.e., LIR$_P$ $>$ LIR$_B$ $>$ LIR, GIR$_P$ $>$ GIR$_B$ $>$ GIR, HIR$_P$ $>$ HIR$_B$ $>$ HIR. This ranking order demonstrates that our proposed PIF architecture that combines the feature-based and fine-tuning based strategies is the most effective architecture, followed by the fine-tuning based strategy, BERT. While the neural-network based models are worse than the former kinds of models.

5.2. Parameters analysis

Next we turn to **RQ3** and conduct a parameter sensitivity analysis of our HIR related models, i.e., HIR, HIR$_B$ and HIR$_P$. Clearly, as shown in Tables 4 and 5, for different datasets, the same model has varied error rates on different orders of magnitude, e.g., HIR on IMDb and Yelp (6.73 *vs* 34.18). To better present the γ effect of the same model on different datasets, we introduce an evaluation metric, Relative Error Rate (RER), which is defined as, given a dataset, the relative improvement ratio of the lowest error rate with regard to the others with different γ*s*. In addition, we examine the performances of these three models in terms of RER by gradually changing the parameters γ from 0 to 1 with an interval 0.1. We plot the RER results of HIR, HIR$_B$ and HIR$_P$ in Fig. 3a, b and c, respectively.

As shown in Fig. 3a, HIR achieves the lowest error rate when $\gamma = 0.5$ on all datasets (except $\gamma = 0.6$ for Yelp dataset), which is 0 in the figure. In addition, the RER of HIR on each dataset decreases consistently when γ varies from 0 to 0.5 (0.6 for Yelp); after that, the RER metric goes up when γ changes from 0.5 (0.6 for Yelp) to 1. The similar phenomena can be found in Fig. 3b and c. HIR$_B$ and HIR$_P$ both achieve the lowest error rate when $\gamma = 0.5$. In addition, the RER of these two models on each dataset first keeps a stable decrease to the lowest point 0 and then increases stably until $\gamma = 1$.

Interestingly, comparing the curve gradient on both sides of $\gamma = 0.5$, we can find that the gradient of the left side is steeper than that of the right side, which indicates that GIR is capable of causing an increase of error rate more easily than LIR. Furthermore, comparison between the identical model on different datasets, demonstrates that the change ranges of RER on IMDb, DBpedia and TREC, is greater than that on Yelp and AG. The phenomena may be due to the differences of statistical characteristics among these datasets, which require further experiments to find potential reasons.

Curiously, we also want to find whether the relation HIR$_P$ $>$ HIR$_B$ $>$ HIR can always keep unchanged when the trade-off parameter γ increases from 0 to 1. Due to the text space, we only select the dataset Yelp as the analytical object, which has the highest error rate among these datasets. We plot the experimental results in Fig. 4. Clearly, as Fig. 4 shows, we can find that the performance of HIR$_P$ is the lowest in terms of error rate, followed by HIR$_B$, and the highest is HIR, when γ increases from 0 to 1. This result is consistent with the previous finding HIR$_P$ $>$ HIR$_B$ $>$ HIR, i.e., the effectiveness of our PIF architecture. On the other hand, it indicates that the effectiveness of our PIF architecture is not sensitive to the trade-off parameter γ.

5.3. Impact of the text length

To answer **RQ4**, we manually group the text according to the text length L, e.g., 0–100, 100–200, ⋯, 900–1000, $>$ 1000. We campare the performance of interaction representation related models, e.g., LIR, GIR, HIR, HIR$_B$ and HIR$_P$, under different settings of text length. We plot this experimental results in Fig. 5

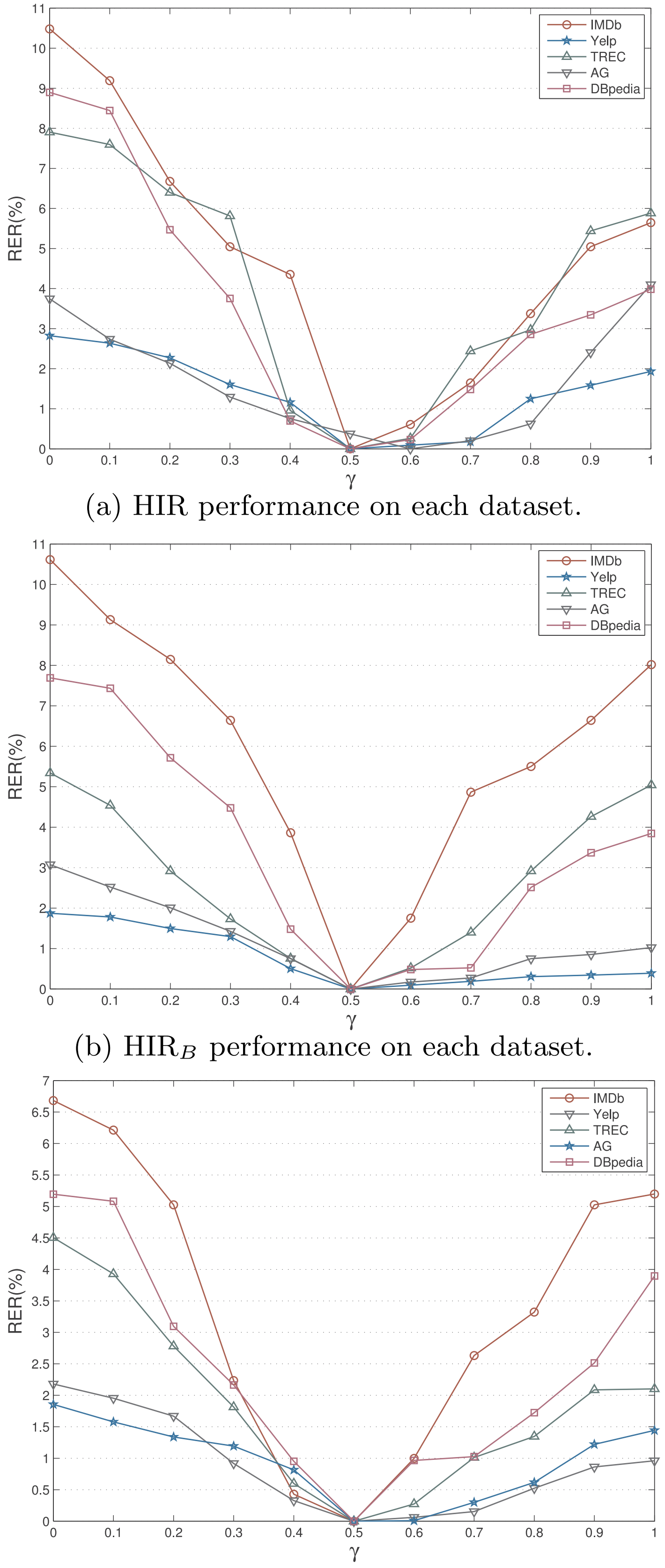
Clearly, as shown in Fig. 5, we can find the relation LIR $>$ GIR $>$ HIR $>$ HIR$_B$ $>$ HIR$_P$ unchanged when text length increases. This phenomenon is consistent with the findings in Section 5.1.1, which indicates the effectiveness of interaction representation and PIF architecture is not affected by the text length.

Interestingly, as the text length increases, the performances of all discussed models decrease first to reach the lowest error rate at the point of group 100–200, and then keep a constant increase. This finding may be explained by the fact that the longer the text, the richer the information it provides, which results in targeting the class label of text more easily, i.e., the decrease of error rate in the earlier stage. But as the text length grows, the structure and semantics of text become more complex and variable, the proposed models find it harder to get the exact representation.

6. Conclusion and future work

In this paper, we focus on the task of text classification and propose a novel pipeline, the PIF architecture, which incorporates the respective advantages from feature based and fine-tuning based strategies in the language model pretrain-finetuning process. We also introduce the concept of interaction representation and propose a two-perspective interaction representation for sentence embeddings, i.e., a local interaction representation (LIR) and a global interaction representation (GIR). We combine these two representations to produce a hybrid interaction representation model, i.e., HIR.

We evaluate the efficiency of these proposed models on five widely-used datasets for text classification. Our experimental results shows that: (1) Compared with the traditional neural-network based models, our basic interaction-related models can help boost the



(a) HIR performance on each dataset.

(b) HIR_B performance on each dataset.

Fig. 3. Effect on performance of the HIR related models in terms of RER by changing the trade-off parameter γ, tested on all datasets.

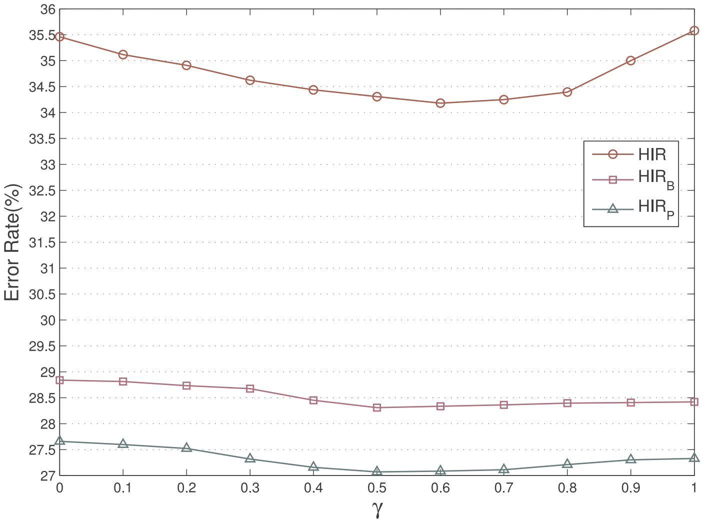


Fig. 4. Effect on the performance of HIR related models in terms of error rate by changing the trade-off parameter γ, tested on the Yelp dataset.

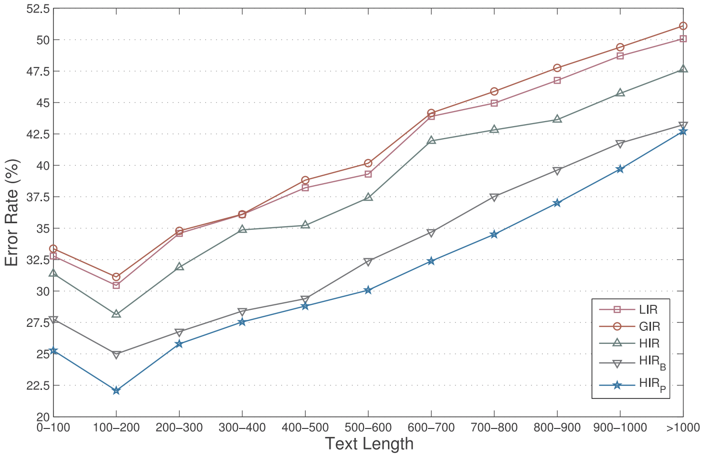


Fig. 5. Effect on the performance of interaction related models in terms of error rate with varied text length, tested on Yelp dataset.

performance for text classification in terms of error rate. (2) Our proposed PIF architecture is more effective to help improve the text classification than the existing feature-based as well as the fine-tuning based strategies. In particular, HIR_P model demonstrates the best performance on each dataset. (3) The effectiveness of interaction representation and the PIF architecture is not affected by the text length.

In future works, we will delve deeper into the realm of 3 aspects : (1) Our proposals are Bi-LSTM based approaches that suffer from the issue of gradient disappearing when dealing with the large text, so we plan to extend our proposal by incorporating some length-robustness models, e.g., the transformer based models. (2) We will evaluate our models for other tasks so as to verify the robustness of the interaction representation models. (3) The existing fine-tuning approaches are too general, so we seek opportunities for further enhancement in the field of task-sensitive fine-tuning methods.

CRediT authorship contribution statement

Jianming Zheng: Conceptualization, Methodology, Validation. **Fei Cai:** Data curation, Writing - original draft. **Honghui Chen:** Investigation, Resources, Validation. **Maarten de Rijke:** Writing - review & editing.

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China under No. 61702526, the Defense Industrial Technology Development Program under No. JCKY2017204B064, the National Advanced Research Project under No. 6141B0801010b, Ahold Delhaize, the Association of Universities in the Netherlands (VSNU), and the Innovation Center for Artificial

Intelligence (ICAI). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.ipm.2020.102215.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2015). *Neural machine translation by jointly learning to align and translate. ICLR*. OpenReview.net.
Bengio, Y., Ducharme, R., & Vincent, P. (2000). *A neural probabilistic language model. NIPS*. Curran Associates, Inc.932–938.
Bernauer, L., Han, E. J., & Sohn, S. Y. (2018). Term discrimination for text search tasks derived from negative binomial distribution. *Information Processing and Management, 54*(3), 370–379.
Dai, A. M., & Le, Q. V. (2015). *Semi-supervised sequence learning. NIPS*. Curran Associates, Inc.3079–3087.
Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding. NAACL-HLT*. Association for Computational Linguistics4171–4186.
He, R., Lee, W. S., Ng, H. T., & Dahlmeier, D. (2018). *Exploiting document knowledge for aspect-level sentiment classification. ACL*. Association for Computational Linguistics579–585.
Howard, J., & Ruder, S. (2018). *Universal language model fine-tuning for text classification. ACL*. Association for Computational Linguistics328–339.
Ioffe, S., & Szegedy, C. (2015). *Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML*. JMLR.org448–456.
Joachims, T. (1998). *Text categorization with support vector machines: Learning with many relevant features. Machine learning: ECML-98*. Springer Berlin Heidelberg137–142.
Johnson, R., & Zhang, T. (2017). *Deep pyramid convolutional neural networks for text categorization. ACL*. Association for Computational Linguistics562–570.
Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). *Bag of tricks for efficient text classification. EACL*. Association for Computational Linguistics427–431.
Kenter, T., Borisov, A., & de Rijke, M. (2016). *Siamese CBOW: Optimizing word embeddings for sentence representations. ACL*. Association for Computational Linguistics941–951.
Kim, Y. (2014). *Convolutional neural networks for sentence classification. EMNLP*. Association for Computational Linguistics1746–1751.
Lai, S., Xu, L., Liu, K., & Jun, Z. (2015). *Recurrent convolutional neural networks for text classification. AAAI*. IEEE2267–2273.
Lewis, D. D. (1992). *An evaluation of phrasal and clustered representations on a text categorization task. SIGIR*. ACM37–50.
Liu, P., Qiu, X., & Huang, X. (2016). *Recurrent neural network for text classification with multi-task learning. IJCAI*. IJCAI/AAAI Press2873–2879.
Luong, M. T., Pham, H., & Manning, C. D. (2015). *Effective approaches to attention-based neural machine translation. EMNLP*. Association for Computational Linguistics1412–1421.
Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). *Learning word vectors for sentiment analysis. ACL*. Association for Computational Linguistics142–150.
Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. (2014). *The stanford corenlp natural language processing toolkit. ACL*. Association for Computational Linguistics55–60.
McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). *Learned in translation: Contextualized word vectors. NIPS*. Curran Associates, Inc.6297–6308.
Mehta, P., & Majumder, P. (2018). Effective aggregation of various summarization techniques. *Information Processing and Management, 54*(2), 145–158.
Merity, S., Keskar, N. S., & Socher, R. (2018). *Regularizing and optimizing LSTM language models. ICLR*. OpenReview.net.
Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space. ICLR*. OpenReview.net.
Parikh, A. P., Täckström, O., Das, D., & Uszkoreit, J. (2016). *A decomposable attention model for natural language inference. EMNLP*. Association for Computational Linguistics2249–2255.
Pascanu, R., Mikolov, T., & Bengio, Y. (2013). *On the difficulty of training recurrent neural networks. ICML*. JMLR.org1310–1318.
Pennington, J., Socher, R., & Manning, C. D. (2014). *Glove: Global vectors for word representation. EMNLP*. Association for Computational Linguistics.
Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). *Deep contextualized word representations. NAACL-HLT*. Association for Computational Linguistics2227–2237.
Post, M., & Bergsma, S. (2013). *Explicit and implicit syntactic features for text classification. ACL*. Association for Computational Linguistics866–872.
Qin, C., Qinmin, H., Jimmy Xiangji, H., Liang, H., & Weijie, A. (2017). *Enhancing recurrent neural networks with positional attention for question answering. SIGIR*. ACM993–996.
Ranzato, M. A., Boureau, Y. L., & Lecun, Y. (2007). *Sparse feature learning for deep belief networks. NIPS*. Curran Associates, Inc.1185–1192.
Ren, P., Chen, Z., Ren, Z., & de Rijke, M. (2017). *Leveraging contextual sentence relations for extractive summarization using a neural attention model. SIGIR*. ACM95–104.
Robertson, S. (2004). Understanding inverse document frequency: On theoretical arguments for IDF. *Journal of Documentation, 60*(5), 503–520.
Saif, H., He, Y., Fernández, M., & Alani, H. (2016). Contextual semantics for sentiment analysis of twitter. *Information Processing and Management, 52*(1), 5–19.
Saraiva, P. C., Cavalcanti, J. M. B., de Moura, E. S., Gonçalves, M. A., & da Silva Torres, R. (2016). A multimodal query expansion based on genetic programming for visually-oriented e-commerce applications. *Information Processing and Management, 52*(5), 783–800.
Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., & Potts, C. (2013). *Recursive deep models for semantic compositionality over a sentiment treebank. EMNLP*. Association for Computational Linguistics1631–1642.
Tai, K. S., Socher, R., & Manning, C. D. (2015). *Improved semantic representations from tree-structured long short-term memory networks. ACL*. Association for Computational Linguistics1556–1566.
Tang, D., Qin, B., & Liu, T. (2015). *Document modeling with gated recurrent neural network for sentiment classification. EMNLP*. Association for Computational Linguistics1422–1432.
Voorhees, E. M., & Tice, D. M. (1999). *The TREC-8 question answering track evaluation. TREC*. National Institute of Standards and Technology (NIST)82.
Xie, H., Li, X., Wang, T., Lau, R. Y. K., Wong, T., Chen, L., ... Li, Q. (2016). Incorporating sentiment into tag-based user profiles and resource profiles for personalized search in folksonomy. *Information Processing and Management, 52*(1), 61–72.
Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR, abs/1906.08237*.
Zhang, X., Zhao, J., & Lecun, Y. (2015). *Character-level convolutional networks for text classification. NIPS*. Curran Associates, Inc.649–657.
Zheng, J., Cai, F., Chen, W., Feng, C., & Chen, H. (2019). Hierarchical neural representation for document classification. *Cognitive Computation, 11*(2), 317–327.
Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., & Xu, B. (2016). *Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. COLING*. Association for Computational Linguistics3485–3495.