Expert Systems With Applications 207 (2022) 117924

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Multiple userids identification with deep learning

Xin Du, Siyuan Chen, Zhiyue Liu, Jiahai Wang *

School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China

ARTICLE INFO

Keywords:
Multiple userid identification
Social media
Document similarity
Deep learning
Deep semantic features

ABSTRACT

People are becoming increasingly active on social media. It is common knowledge that most users register more than one social media account. Some users use multiple accounts to publish fake reviews to disorder the regular social network. To identify multiple accounts of the same author, multiple userids identification problem is defined. Existing methods for multiple userids identification problem focus on human-designed shallow statistical features and do not fully utilize the deep semantic information in user-generated texts. This paper uses the deep learning model to extract deep semantic features at the document level and user level. Then the most similar user pairs can be selected by computing the similarity of their writing styles indicated by text features. The experimental results demonstrate that our deep learning method outperforms state-of-the-art methods.

1. Introduction

Social networks are widely employed in our daily lives and thus, have exerted an enormous influence on our social lives. Social networks allow people to share their information, knowledge and interests with ease and novel enthusiasm. As a result, many social computing tasks/applications arise, such as public opinion analysis and recommender systems based on social networks (Hu, Wang, Ren, & Choo, 2018; Jain, Katarya, & Sachdeva, 2020; Mohammadmosaferi & Naderi, 2020). However, some people use convenience to create multiple userids to perform vicious acts. They use these accounts to write fake reviews and attack their opponents. Even worse, some doppelgängers may exist in certain underground forums. They form a cybercrime ecosystem to exchange illicit goods and services (Afroz, Islam, Stolerman, Greenstadt, & Mccoy, 2014). Therefore, identifying these accounts and banning their behaviors will purify our social network environment. A problem is proposed based on this situation. This problem is referred to as multiple userids identification problem (MUIP) (Kim, Noh, & Park, 2015; Qian & Liu, 2013). It can be defined as follows: Assume that there is a set of authors $Author = \{a_1, a_2, \dots, a_m\}$ and a set of userids $ID = \{id_1, id_2, \dots, id_n\}$. Each author has two or more userids. Each id_i has a set of documents $D_i = \{D_{i1}, D_{i2}, \dots\}$. Our task is to identify userids that belong to the same author by their documents, because everyone has a personal specific writing style, such a tendency to use singular first-person pronouns (e.g., I) or to use fewer negation words. This problem is fundamentally important in social network information diffusion and user behavior analysis. The detection of sockpuppets in Wikipedia (Solorio, Hasan, & Mizan, 2013) is a good example. Sockpuppets are fake accounts created by malicious users to disorder the regulations of Wikipedia. These accounts can register multiple identities for various purposes, including false majority opinion claims, block evasion, and vote stacking, but Wikipedia does not have a proper method to detect them. Recently, an increasing number of works have concentrated on spam review detection and spammer detection (Fahfouh, Riffi, Adnane Mahraz, Yahyaouy, & Tairi, 2020; Manaskasemsak, Tantisuwankul, & Rungsawang, 2021). They have designed deep learning models to distinguish fake reviewers from benign ones. However, some fake accounts are controlled by the same author such as puppetmaster (Kumar, Cheng, Leskovec, & Subrahmanian, 2017). These puppetmasters will create more accounts to constantly publish fake reviews. Thus, identifying these accounts is urgently needed.

Some methods (Kim et al., 2015; Qian & Liu, 2013) have been proposed to solve MUIP. However, these methods require labor-intensive feature engineering. They encode input text pairs using many complex lexical and syntactic features. Moreover, these methods extract manually crafted features without considering latent semantic features. Although some methods, such as learning in the similarity space (LSS) (Qian & Liu, 2013), extract rich manually crafted features, they only pay attention to stand-alone words without considering the relationship of contextual words. Therefore, these methods can hardly grasp the latent semantic information of texts. To address the shortcomings of existing works on MUIP, a deep learning method is proposed in this paper to extract latent/deep semantic features for MUIP.

* Corresponding author.
E-mail addresses: duxin23@mail2.sysu.edu.cn (X. Du), chensy47@mail2.sysu.edu.cn (S. Chen), liuzhy93@mail2.sysu.edu.cn (Z. Liu), wangjiah@mail.sysu.edu.cn (J. Wang).

https://doi.org/10.1016/j.eswa.2022.117924
Received 14 February 2020; Received in revised form 17 June 2022; Accepted 18 June 2022
Available online 27 June 2022

This paper proposes a deep learning method that is referred to as learning document-level and user-level features with deep learning (DU-DL) to solve MUIP. This method learns user writing style from two aspects: document level and user level. At the document level, a single document reflects its own writing style, while all documents of a user reflect the global writing style at the user level. DU-DL is composed of two stages: deep semantic feature extraction and candidate user identification. In the deep semantic feature extraction stage, a deep learning model is proposed to extract and learn document-level features and user-level features. In the candidate user identification stage, document-level and user-level features are combined to identify the most similar user pairs. The main contributions of this paper are threefold:

- A two-stage deep learning method DU-DL is proposed to solve MUIP. DU-DL extracts deep semantic features from two aspects: document level and user level. Therefore, DU-DL enhances the ability to learn user writing style.
- DU-DL combines manually crafted feature extraction methods and the deep learning method to extract features. Based on the reprocessing of manually crafted feature extraction methods, the deep learning method can calculate the user similarity score by using all documents of the user. Therefore, DU-DL can calculate the user similarity score from the global perspective.
- DU-DL is evaluated on two different real-world datasets, Weibo and Douban. The experimental results show the superiority of DU-DL.

The remaining sections are organized as follows. Section 2 briefly reviews related works. Section 3 introduces DU-DL. Section 4 presents the experimental setups and experimental results. Section 5 concludes this paper.

2. Related works

The proposed method DU-DL aims to identify multiple userids by using the sentence pair model based on deep learning. Therefore, this section reviews existing works on MUIP, deep learning and sentence pair models.

2.1. Existing works on MUIP

To identify multiple userids of the same author, Chen, Goldberg, and Magdonismail (2004) proposed a model based on the timing sequence law of posts. This model does not use any linguistic clues. Therefore, it is not applicable to domains such as online texts. Novak, Raghavan, and Tomkins (2004) employed a clustering-based method, but the exact number of users cannot be known. This is unrealistic in practice.

Recent works focus on MUIP by learning user writing styles. Qian and Liu (2013) proposed a method that is referred to as learning in the similarity space (LSS). LSS expends the text classification problem from the original document space to the similarity space. LSS extracts rich manually crafted features to learn user writing style, such as term frequency (TF), term frequency and inverse document frequency (TF-IDF) (Jones, 1972) and *n*-gram. Kim et al. (2015) proposed a method extracting high-dimensional text features from Korean texts. Afroz et al. (2014) proposed a doppelgänger finder model to detect multiple accounts of the same author in underground forums. This model uses principal components analysis (PCA) to extract principal components of *n*-gram features. These features are used to calculate the pairwise probability of two authors. Some works define the concept of sockpuppets, which are user accounts created by the same author. Kumar et al. (2017) divided social network users into ordinary users and sockpuppets. This method uses activity features, community features and post features to identify sockpuppets. Hosseinia and Mukherjee (2017) applied KL divergence on stylistic language models to find discriminative features. To detect sockpuppets from test samples, this method uses nearest and farthest neighbors to retrieve hidden samples by sending a set of positive samples from the training set in the unlabeled test set. Yamak (2018) utilized a SocksCatch model to detect multiple userids. This model firstly uses machine learning algorithms to detect the sockpuppet accounts. Then, SocksCatch uses community detection algorithms to group sockpuppet accounts created by the same author. Khoory, Al Abdooli, Al Roken, and Hacid (2019) developed a tool named Speculo to detect sockspuppets in Twitter. These existing methods only extract manually crafted features from user documents. This paper proposes a two-stage deep learning method DU-DL to solve MUIP. This method extracts deep semantic features at the document level and user level to identify multiple userids of the same author. In addition, manually crafted feature extraction methods are combined with the deep learning method in DU-DL. Based on the reprocessing of manually crafted feature extraction methods, DU-DL can more effectively extract deep semantic features.

2.2. Deep learning

Deep learning allows computational models that consist of multiple processing layers to automatically learn intricate representations of input data (LeCun, Bengio, & Hinton, 2015). Deep learning has been demonstrated to be effective in discovering certain nonlinear, complex structures in high-dimensional data, such as spatial structure and temporal structure in abstractive summary (Chen et al., 2022), personality detection (Ren, Shen, Diao, & Xu, 2021; Yang, Quan, Yang, & Yu, 2021) and spam detection (Alharthi, Alhothali, & Moria, 2021; Cao, Ji, Chiu, & Gong, 2022). Certain specialized deep learning models can take advantage of these structures, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs) and stacked autoencoders (SAEs) (Rouast, Adam, & Chiong, 2021).

CNNs have achieved great successes in processing images and video by learning features from the spatial structures of the data (LeCun et al., 2015). The spatial structures are taken into account by several learnable kernels in CNNs. The same kernel can be applied over different spatial locations. This kernel is referred to as parameter sharing among units. The receptive field of the kernel is much smaller than the input. This characteristic leads to sparse connectivity between units of adjacent layers (Rouast et al., 2021).

RNNs are widely employed in processing texts and speeches by learning sequential information (LeCun et al., 2015). RNNs extend the feedforward network by allowing recurrent connections to exist within layers. Namely, the outputs of hidden states can be regarded as additional inputs at each temporal step. Therefore, RNNs can form memories in the hidden states over information from all previous inputs (Ming et al., 2017). The most relevant variants of RNNs are long short-term memory (LSTM) and gated recurrent unit (GRU). LSTM is successful at learning long-term dependencies by using gate mechanisms to selectively add and forget information (Hochreiter & Schmidhuber, 1997). GRU is a gating mechanism that is proposed in the context of sequence-to-sequence processing (Cho, van Merrienboer, Gülçehre, Bougares, Schwenk et al., 2014).

SAEs are a type of unsupervised deep learning models. This type of deep learning models can be utilized to learn low-dimensional feature representations from input data. The low-dimensional feature representation can be further applied to other tasks.

CNNs and RNNs are widely employed in natural language processing (NLP) tasks (Li, 2018; Young, Hazarika, Poria, & Cambria, 2018). They can be applied to learn latent semantic feature representations of text or speech. On several NLP tasks, the performances of these deep learning models outperform traditional shallow models, such as support vector machines and logistic regression. Therefore, deep learning models can be utilized in this paper to extract text features and compare the writing styles of different authors.

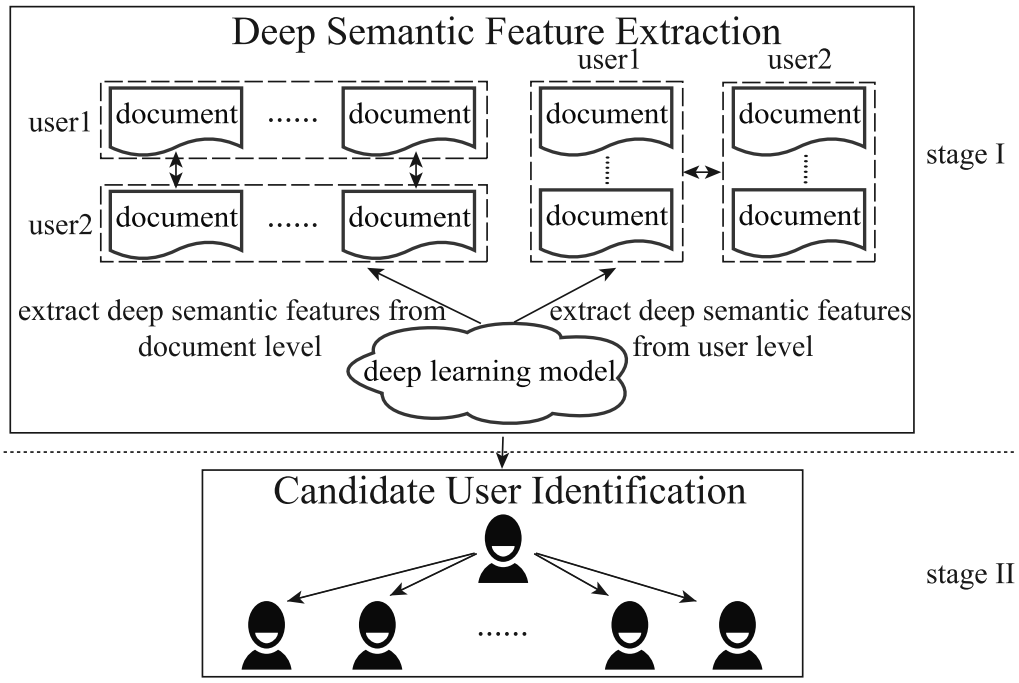


Fig. 1. Framework of DU-DL. In the first stage, a deep learning model is used to extract document-level and user-level features. In the second stage, document-level and user-level features are combined to calculate the similarity between two userids. The most similar user pairs can be obtained.

2.3. Sentence pair models

When deep learning models are applied to NLP tasks, the input data can be one or two sentences. When the input data consist of two sentences, this type of deep learning models can be referred to as sentence pair models (Lan & Xu, 2018). Sentence pair models aim to calculate the similarity between two sentences and to capture the relationship. Therefore, sentence pair models can be used to identify multiple userids of the same author by comparing their writing styles. There are two types of sentence pair models: sentence encoding models and sentence pair interaction models (Lan & Xu, 2018).

For sentence encoding models, Shen et al. (2018) proposed the simple word-embedding model. This model uses simple pooling operations to extract features on different tasks, such as sentiment analysis and answer sentence selection. The model achieves similar or even superior results on these tasks compared with CNNs and LSTM. Conneau, Kiela, Schwenk, Barrault, and Bordes (2017) proposed the Bi-LSTM max-pooling network model (InferSent). This model uses bidirectional LSTM to encode sequential contexts. Nie and Bansal (2017) proposed the short-stacked sentence encoder model. This model uses a 3-layer Bi-LSTM compared with InferSent. These sentence encoding models focus on the vector representation of individual sentences. RNNs are usually employed in these models.

For sentence pair interaction models, He and Lin (2016) proposed the pairwise word interaction model (PWIM). The innovation of this model is that it creates a 13-layer tensor. This tensor includes cosine similarity, Euclidean distance and dot product information over the outputs of the previous encoding layer. Parikh, Täckström, Das, and Uszkoreit (2016) proposed the decomposable attention model (DecAtt). This model is one of the earliest models to introduce attention-based alignment for sentence pair modeling (Lan & Xu, 2018). Chen, Zhu, Ling, Wei, Jiang et al. (2016) proposed the enhanced sequence inference model (ESIM). This model adds Tree-LSTM based on DecAtt. These sentence pair interaction models focus on the relationship between two sentences and aggregate intersentence interactions (Lan & Xu, 2018). Due to these characteristics, this paper selects the sentence pair interaction model as the main component in DU-DL.

3. Proposed method

The framework of DU-DL is presented in Fig. 1. DU-DL works in two stages. In the first stage, deep semantic features, including document-level and user-level features, are extracted by a deep learning model. In the second stage, document-level and user-level features are combined to calculate the similarity between two userids. The most similar user pairs can be obtained. The details of the framework are given as follows.

3.1. Deep semantic feature extraction

In the deep semantic feature extraction stage, the sentence pair interaction model DecAtt is introduced for feature extraction. This model is then applied to learn deep semantic features at the document level and user level.

3.1.1. Deep learning model: DecAtt

DecAtt (Parikh et al., 2016) is selected to evaluate the semantic closeness between two different documents in this stage. The reason is that DecAtt is representative of sentence pair interaction models. DecAtt achieves state-of-the-art results on the SNLI dataset (Bowman, Angeli, Potts, & Manning, 2015), which is a popular sentence pair dataset. In addition, DecAtt has fewer parameters to be trained and does not rely on word order information (Lan & Xu, 2018) compared with other sentence pair interaction models. Therefore, DecAtt is adopted in DU-DL.

The architecture of DecAtt is presented in Fig. 2. DecAtt has six components: word embedding layer, soft-alignment layer, comparison layer, summation layer, concatenation layer and multilayer perceptron (MLP) layer. At the word embedding layer, each word is mapped to a vector by using pretrained word embeddings. Therefore, an input sentence pair can be mapped to two sentence matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$. At the soft-alignment layer, an attention matrix $\boldsymbol{W}$ is built by multiplying the two sentence matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$. In the attention matrix, the greater the similarity in the semantics of two words, the more the cross section of the two words towards white. The two sentence matrices can be aligned by this attention matrix. Then two aligned phrases $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ can

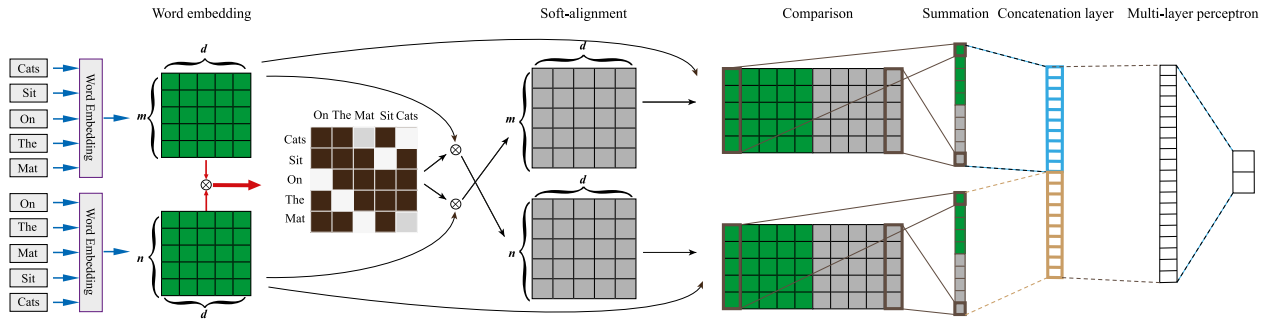


Fig. 2. The sentence pair interaction model DecAtt consists of six components: word embedding layer, soft-alignment layer, comparison layer, summation layer, concatenation layer and multilayer perceptron layer. m and n are the word counts of two sentences; d is the word vector dimension; and $\otimes$ is the matrix multiplication symbol.

be generated. At the comparison layer, the aligned phrases $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are concatenated with $\boldsymbol{Y}$ and $\boldsymbol{X}$, respectively, for comparison. At the summation layer, the same dimension data of each row of the concatenation matrix are added. Two concatenation vectors $\boldsymbol{s}^1$ and $\boldsymbol{s}^2$ can be generated. At the concatenation layer, $\boldsymbol{s}^1$ and $\boldsymbol{s}^2$ are concatenated into one feature vector $\boldsymbol{c}$. The MLP is applied to perform binary-class classification for the feature vector $\boldsymbol{c}$. Each component is described in detail as follows:

- **Word Embedding**: This component encodes each word to its corresponding vector representation. Therefore, an input sentence pair can be mapped to two sentence matrices $\boldsymbol{X} \in \mathbb{R}^{m\times d}$ and $\boldsymbol{Y} \in \mathbb{R}^{n\times d}$, where m and n are the word counts of two sentences; and d is the dimension of word vectors.
- **Soft-Alignment**: This component generates two aligned phrases $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ corresponding to $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. Firstly, an attention matrix $\boldsymbol{W}$ is built by multiplying the two sentence matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$. Each weight W_{ij} in $\boldsymbol{W}$ represents the correlation degree between word i in $\boldsymbol{X}$ and word j in $\boldsymbol{Y}$. The larger W_{ij} is, the greater similar between word i and word j. Then, the elements of $\boldsymbol{X}$ and $\boldsymbol{Y}$ are aligned by using the attention matrix $\boldsymbol{W}$. Finally, two aligned phrases $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ are generated. Similar operations can be seen in pointer networks (Vinyals, Fortunato, & Jaitly, 2015) and BERT (Devlin, Chang, Lee, & Toutanova, 2019). The process is presented as follows:

$$\boldsymbol{W} = \boldsymbol{X}\boldsymbol{Y}^T, \tag{1}$$

$$\boldsymbol{\alpha}_j = \sum_{i=1}^{m} \frac{exp(W_{ij})}{\sum_{a=1}^{m} exp(W_{aj})} \cdot \boldsymbol{x}_i \qquad \forall j \in [1, \ldots, n], \tag{2}$$

$$\boldsymbol{\beta}_i = \sum_{j=1}^{n} \frac{exp(W_{ij})}{\sum_{a=1}^{n} exp(W_{ia})} \cdot \boldsymbol{y}_j \qquad \forall i \in [1, \ldots, m], \tag{3}$$

  where $\boldsymbol{x}_i$ denotes the ith row of $\boldsymbol{X}$; $\boldsymbol{y}_j$ denotes the jth row of $\boldsymbol{Y}$; W_{ij} denotes an element of $\boldsymbol{W} \in \mathbb{R}^{m\times n}$; $\boldsymbol{\alpha} \in \mathbb{R}^{n\times d}$; $\boldsymbol{\beta} \in \mathbb{R}^{m\times d}$;
- **Comparison**: This component compares $\boldsymbol{x}_i$ with $\boldsymbol{\beta}_i$ and $\boldsymbol{y}_i$ with $\boldsymbol{\alpha}_i$ by concatenating them. The process is presented as follows:

$$\boldsymbol{v}_i^1 = G([\boldsymbol{x}_i, \boldsymbol{\beta}_i]), \tag{4}$$

$$\boldsymbol{v}_i^2 = G([\boldsymbol{y}_i, \boldsymbol{\alpha}_i]), \tag{5}$$

  where $G([x, y])$ denotes a function to learn the relationship between x and y. Here, $[\cdot]$ denotes the concatenation operation symbol; $G(\cdot)$ is a feed-forward network. This network maps vectors from the $2d$ dimension to the d dimension. Therefore, $\boldsymbol{v}^1 \in \mathbb{R}^{m\times d}$ and $\boldsymbol{v}^2 \in \mathbb{R}^{n\times d}$. The superscript number is the index of two sentences.
- **Summation**: This component aggregates $\boldsymbol{v}^1$ and $\boldsymbol{v}^2$. The summation operation is presented as follows:

$$\boldsymbol{s}_j^1 = \sum_{i=1}^{m} \boldsymbol{v}_{ij}^1, \tag{6}$$

$$\boldsymbol{s}_j^2 = \sum_{i=1}^{n} \boldsymbol{v}_{ij}^2, \tag{7}$$

  where $\boldsymbol{s}^1, \boldsymbol{s}^2 \in \mathbb{R}^d$.
- **Concatenation**: This component concatenates $\boldsymbol{s}^1$ and $\boldsymbol{s}^2$:

$$\boldsymbol{c} = [\boldsymbol{s}^1, \boldsymbol{s}^2]. \tag{8}$$

- **Multi-Layer Perceptron**: This classifier consists of two hidden layers with an $ReLU$ activation function and with a softmax output layer. The outputs of both hidden layers are d-dimensional feature vectors. Softmax calculates the probability for the ith category as follows:

$$p(y = i|\boldsymbol{c}; \boldsymbol{\theta}) = \frac{e^{\boldsymbol{c}^T\boldsymbol{\theta}_i}}{\sum_{k=1}^{K} e^{\boldsymbol{c}^T\boldsymbol{\theta}_k}}, \tag{9}$$

  where y is the class label; $\boldsymbol{c}$ is the input vector; $\boldsymbol{\theta}_k$ is a weight vector of the kth class; and K is the count of the classes. This paper uses MLP as the binary-class classifier.

3.1.2. Document-level feature extraction

Document-level feature extraction is proposed to extract deep semantic features from each document pair. The document pairs can be generated as follows.

For two userids u_q and u_s, the two document sets Q and S belong to u_q and u_s, respectively. Assume that there are two documents $Q_i \in Q$ and $S_j \in S$. They can form a document pair $< Q_i, S_j >$. $< Q_i, S_j >$ is referred to as a positive sample if u_q and u_s belong to the same author; otherwise, it is referred to as a negative sample. The deep learning model can take these samples as inputs to extract document-level features.

3.1.3. User-level feature extraction

User-level feature extraction is proposed to extract deep semantic features from the keywords of each user. The keywords can be extracted from the document set of the user. For example, a user has 10 documents. A dictionary can be generated by word frequency statistics in the 10 documents. Then top-100 words can be selected from the 10 documents to present the frequently used words of the user. Here, the top-100 words are keywords of the user. Finally, the similarity of two users can be computed by comparing their keywords.

Keywords can be extracted by three strategies, TF, TF-IDF and TextRank (Mihalcea & Tarau, 2004). TF is the most commonly employed feature. It presents the word frequency distribution of a document. TF-IDF is similar to TF. The difference is that TF-IDF reduces the weights of frequent words by the idf function. idf function and TF-IDF are defined as follows:

$$idf(t, D) = log\frac{N}{|\{d \in D : t \in d\}|}, \tag{10}$$

$$tf\text{-}idf(t, D) = tf(t, D) \cdot idf(t, D), \tag{11}$$

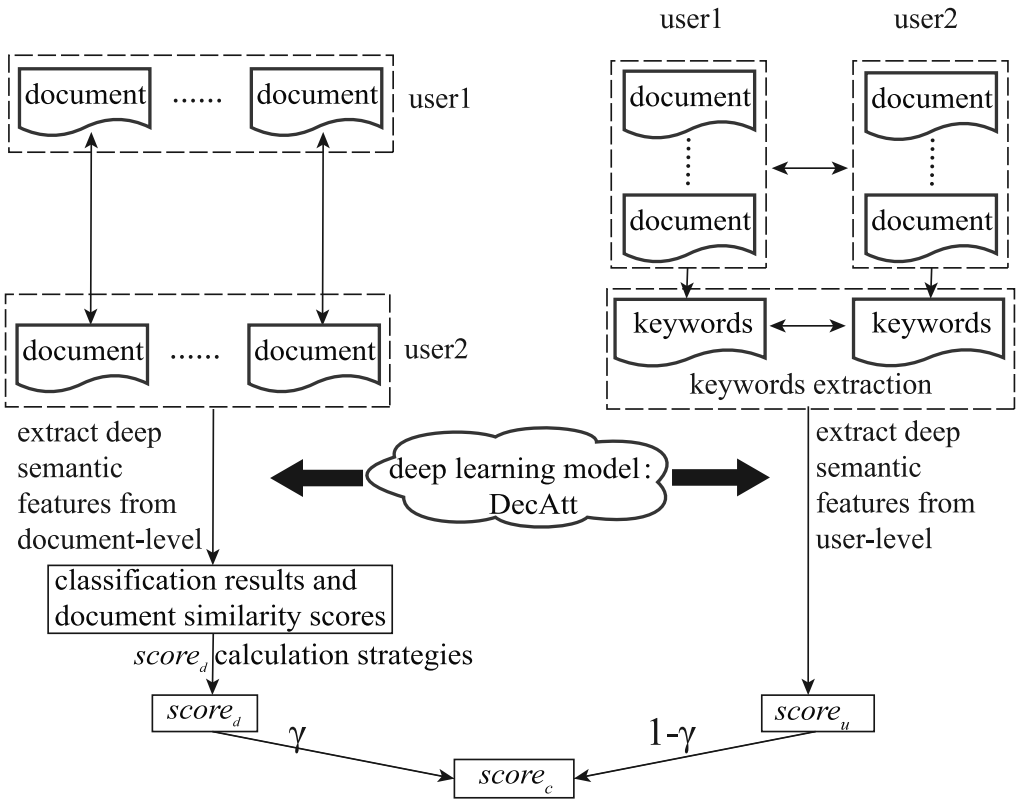


Fig. 3. Process of the user similarity score calculation. $score_d$ and $score_u$ show the user similarity from the document level and user level, respectively. $score_c$ is the final user similarity score.

where t is a word term; D is the document set; N is the count of documents; tf is a function that computes the frequency of occurrence of a word term; and $|\{d \in D : t \in d\}|$ represents the number of documents where the word term t appears (i.e., $tf(t, D) \neq 0$). From the definition of the idf function, the more frequent a word appears, the less important the word is.

The TextRank algorithm is similar to PageRank (Page, 1998), which is the most popular algorithm to rank the web page. TextRank uses the word co-occurrence principle to measure the weight of the word and extract keywords of the document. The algorithm description is presented as follows:

$$WS(V_i) = (1-d) + d \cdot \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} \cdot WS(V_j), \tag{12}$$

where $WS(V_i)$ is the importance score of the vertex V_i; $d \in [0, 1]$; $In(V_i)$ is the set of vertices that point to V_i (predecessors); V_j points to each vertex of $Out(V_j)$ (successors); and w_{ji} is the weight of the edge from V_j to V_i. Here, a word is considered a vertex. Therefore, the importance of a vertex V_i depends on the count of the vertexes that point to V_i.

After the keywords of each user are extracted, the deep learning model can extract user-level features from the keyword sets. For two userids u_q and u_s, the document sets Q and S belong to u_q and u_s, respectively. Two keyword sets W_q and W_s are extracted from Q and S, respectively. $< W_q, W_s >$ can be a keyword set pair. $< W_q, W_s >$ is referred to as a positive sample if u_q and u_s belong to the same author; otherwise, it is referred to as a negative sample. The deep learning model can apply these samples as inputs to extract user-level features.

3.1.4. Training

DecAtt is trained by using document pairs indicated in document-level feature extraction. The reasons are twofold: (1) Compared with the unordered keywords of each user, the semantic diversity of the contexts in the document set can be learned by the deep learning model. (2) The number of samples constituted by keyword set pairs is substantially less than that of document pairs. Therefore, the samples constituted by the document pairs are selected as the training set for DecAtt.

To construct the training set, assume that there is a set of authors $AR = \{a_1, a_2, \ldots, a_m\}$. Each author a_i has a set of documents $D_i = \{D_{i1}, D_{i2}, \ldots\}$. For each author a_i, its document set D_i can be split into two subsets $sub\_D_i^1$ and $sub\_D_i^2$. For two arbitrary subsets $sub\_D_i^1$ and $sub\_D_j^2$, there are two documents $dr_1 \in sub\_D_i^1$ and $dr_2 \in sub\_D_j^2$. The two documents can form a document pair $< dr_1, dr_2 >$. When $i = j$, $< dr_1, dr_2 >$ can be a positive sample of the training set. Otherwise, $< dr_1, dr_2 >$ can be a negative sample of the training set. The number of negative samples is substantially greater than that of positive samples. In case of imbalance in the training process, only a part of the negative samples can be randomly selected during the training process.

After the training set is constructed, the training of DecAtt can be started. Cross-entropy (De Boer, Kroese, Mannor, & Rubinstein, 2005) is utilized as the loss function in DecAtt. The loss function is defined as follows:

$$loss = -\sum_i [y_i log \hat{y}_i + (1-y_i) log(1-\hat{y}_i)] + \lambda \|\theta\|^2, \tag{13}$$

where y_i is the class label for the ith sample; $\hat{y}_i$ is the predicted distribution; λ is the L_2-regularization term; and θ is the parameter set. AdaDelta (Zeiler, 2012) is selected as the optimizer to minimize the cross-entropy cost function. The reason is that AdaDelta dynamically updates parameters over time using only first-order information and has less computational cost than the stochastic gradient descent method.

3.2. Candidate user identification

Now, the second stage of DU-DL, candidate user identification, can be performed. This paper randomly splits the document set of author

a_i into query set Q_i and sample set S_i. It can be defined that Q_i and S_i belong to a query userid uq_i and a sample userid us_i, respectively. This stage aims to identify the pairs of uq_i and us_i that belong to the same author.

To identify the pairs, the similarity score should be calculated between the query userid and the sample userid. Document-level feature extraction and user-level feature extraction can be performed to calculate two user similarity scores $score_d$ and $score_u$, respectively. $score_d$ and $score_u$ are defined as follows:

$score_d$ can be calculated via two strategies, Votes and Sums.

- **Votes**: Each document pair is either predicted as positive or negative by the classifier. This strategy computes the percentage of the predicted positive document pairs. $score_d$ of each user pair can be calculated as follows:

$$score_d = \frac{\#\text{positive samples}}{\#\text{document pairs}}. \tag{14}$$

- **Sum**: This strategy averages the scores of all the predicted document pairs. The calculation process is presented as follows:

$$score_d = \frac{\text{the summation of classification scores}}{\#\text{document pairs}}. \tag{15}$$

$score_u$ can be calculated by using user keywords. Assume that there are two top-n keyword sets $\{W_t^1\}_{t=1}^n$ and $\{W_t^2\}_{t=1}^n$ from two different users. $score_u$ is calculated as follows:

$$score_u = F(\{W_t^1\}_{t=1}^n, \{W_t^2\}_{t=1}^n), \tag{16}$$

where F is a function that can calculate the similarity between two word sequences. Here, the deep learning model DecAtt acts as the function F.

Next, a final user similarity score $score_c$ can be calculated by combining $score_d$ with $score_u$ using the weight γ, which is calculated as follows:

$$score_c = \gamma \times score_d + (1-\gamma) \times score_u. \tag{17}$$

The process of the user similarity $score_c$ calculation can be summarized and shown in Fig. 3. In the process of document-level feature extraction, the deep learning model DecAtt extracts deep semantic features of texts and generates the classification results of the document pairs. Votes or Sum can use the classification results to calculate the user similarity score $score_d$. In the process of user-level feature extraction, the keywords of each user can be extracted by using TF, TF-IDF or TextRank. DecAtt can use these keywords as inputs to calculate $score_u$. Finally, a final user similarity score $score_c$ can be calculated by combining $score_d$ with $score_u$.

After the $score_c$ of each user pair is calculated, the query userid uq_i is matched with the closest sample userid us_j. Similarly, the sample userid us_j is matched with the closest query userid uq_k. us_j is the candidate user for uq_i if $i = k$. Otherwise, there is no candidate user for uq_i.

3.3. Procedure of the proposed method: DU-DL

The algorithm process of DU-DL is presented in Algorithm 1. In the first stage, DecAtt is applied to extract deep semantic features from the document level and user level. The difference between the two feature extraction levels is that they use different input data types. The process of document-level feature extraction uses document pairs. The process of user-level feature extraction uses the pair of keyword sets. Therefore, DecAtt extracts document-level features and user-level features by using different input data. In the second stage, document-level features and user-level features are employed to calculate two user similarity scores $score_d$ and $score_u$, respectively. The two scores can be combined to obtain the most similar user pairs.

The characteristics of DU-DL can be summarized as follows:

- DU-DL combines manually crafted feature extraction methods and the deep learning method to extract features. Manually crafted feature extraction methods can extract keyword sets from user documents. Each keyword set represents the writing style of a specific user. The deep learning model can calculate the similarity score of different keyword sets. This score shows that the user similarity can be calculated by using all documents of the user. Therefore, DU-DL can calculate the user similarity score from the global perspective.
- DU-DL is the first method to integrate document-level and user-level features to solve MUIP. At the document level, a single document reflects the writing style of its own, while all documents of a user reflect global writing style at the user level. Therefore, DU-DL combines document-level and user-level features to improve prediction results.
- DU-DL uses attention mechanism to learn the correlations between two sentence representations. However, many other deep learning methods such as InferSent (Conneau et al., 2017), PWIM (He & Lin, 2016) and ESIM (Chen et al., 2016) design abundant components to learn the feature representations of sentences. They do not pay attention to the intersections between two sentences.
- DU-DL uses DecAtt (Parikh et al., 2016) rather than other deep learning methods to extract deep semantic features, because as mentioned in (Lan & Xu, 2018), DecAtt has fewer parameters to be trained and does not rely on word order information compared with other sentence pair interaction models.

4. Experiment

4.1. Experimental setup

4.1.1. Datasets

Our method is evaluated on two real-world datasets, Weibo and Douban.

- **Weibo**[1]: Weibo is one of the most popular Chinese microblogging websites with approximately 50 million daily active users. Similar to Twitter, Weibo users can publish original tweets, retweet tweets, or mention other users by using "@".
- **Douban**[2]: Douban is a Chinese social networking service website. It allows registered users to record information and create content related to films, books, music, recent events, and activities in Chinese cities. Douban has approximately 200 million registered users as of 2019.

Both datasets include 1970 authors and their online texts. According to (Wang, Lu, Li, & Chen, 2013), several criteria are established to clean the data. Firstly, the original texts are selected as the experimental data but not the retweeted texts. Secondly, the words of each text must be at least 10. Finally, some keywords are set to filter potential advertisements.

For text reprocessing, pyltp (Che, Li, & Liu, 2010) released by Harbin Institute of Technology is selected as a tokenizer. This package tool can perform Chinese word segmentation (Zhang, Deng, Che, & Liu, 2012) and part-of-speech tagging (Li, Zhang, Che, Liu, Chen et al., 2011; Wang, Che, & Liu, 2009) tasks, etc. For word embedding, a pre-trained model[3] trained by Word2Vec is used. This model contains 0.5 million 60-dimensional word vectors, which are fixed during training. Zero vectors are applied to represent out-of-vocab words.

This paper splits the document set of each author into two parts as two user accounts (Qian & Liu, 2013). For the Weibo dataset, 961

[1] Website: http://weibo.com.
[2] Website: http://www.douban.com.
[3] Downloaded from http://pan.baidu.com/s/1boPm2x5.

Algorithm 1: Learning document-level and user-level features with deep learning (DU-DL)

Input: two author sets $QueryUserids = \{uq_1, uq_2, ..., uq_n\}$ and $SampleUserids = \{us_1, us_2, ..., us_n\}$, and the corresponding document sets $QuerySet = \{Q_1, Q_2, ..., Q_n\}$ and $SampleSet = \{S_1, S_2, ..., S_n\}$, the weight γ, keywords count gc

Output: the corresponding $SampleUserids$ index sequence sl for $QueryUserids$

begin
  initialize two $n \times n$ matrices $votes$, sum;
  /* Stage I: Deep Semantic Feature Extraction (Section 3.1) */
  for $i = 1 : n$ **do**
    for $j = 1 : n$ **do**
      for $d_1 = 1 : |Q_i|$ **do**
        for $d_2 = 1 : |S_j|$ **do**
          /* Document-Level Feature Extraction (Section 3.1.2) */
          calculate the similarity score of the document pair $< Q_{id_1}, S_{jd_2} >$ and ss is the result;
          if $ss > 0.5$ **then**
            $votes[i][j] = votes[i][j] + 1$;
          end
          $sum[i][j] = sum[i][j] + ss$;
        end
      end
      $votes[i][j] = \frac{votes[i][j]}{|Q_i| \times |S_j|}$;
      $sum[i][j] = \frac{sum[i][j]}{|Q_i| \times |S_j|}$;
      /* User-Level Feature Extraction (Section 3.1.3) */
      extract keywords $\{W_t^1\}_{t=1}^{gc}$ and $\{W_t^2\}_{t=1}^{gc}$ from Q_i and S_j, respectively;
    end
  end
  /* Stage II: Candidate User Identification (Section 3.2) */
  select a strategy from $votes$, sum to calculate $score_d$;
  calculate the similarity score of each keywords pair $< \{W_t^1\}_{t=1}^{gc}, \{W_t^2\}_{t=1}^{gc} >$ and the result can be seen as $score_u$;
  initialize $n \times n$ matrix $score_c$;
  for $i = 1 : n$ **do**
    for $j = 1 : n$ **do**
      $score_c[i][j] = \gamma \times score_d[i][j] + (1 - \gamma) \times score_u[i][j]$;
    end
  end
  initialize sample userid list sl;
  for $i = 1 : n$ **do**
    $index_1 = \text{argmax}(score_c[i])$;
    $index_2 = \text{argmax}(score_c^T[index_1])$;
    if $index_2 == i$ **then**
      $sl[i] = index_1$;
    end
  end
  return sl;
end

authors are selected, including 861 authors for training and 100 authors for testing. For the Douban dataset, 715 authors are selected, including 615 authors for training and 100 authors for testing. Each author posts at least 100 blogs.

During the training of DecAtt, one document D_{ij} of one author is randomly selected to match the remaining documents of this author. These document pairs can constitute positive samples. The documents of other authors can match D_{ij} to constitute negative samples.

4.1.2. Baseline methods

DU-DL is compared with five baseline methods, including two supervised methods, LSS (Qian & Liu, 2013) and high-dimension LSS (H-LSS) (Kim et al., 2015), two unsupervised methods, TF-C (Qian & Liu, 2013) and TF-IDF-C (Qian & Liu, 2013), and a variant version of DU-DL, DU-DL-I.

- **LSS** (Qian & Liu, 2013): LSS uses SVM to calculate a similarity score for each similarity vector. Each dimension of the similarity vector represents the similarity of two documents from a specific aspect, such as TF and TF-IDF. It obtains state-of-the-art results on a review set from Amazon.com. This paper uses all types of d-features mentioned in (Qian & Liu, 2013). These d-features are length, frequency, TF-IDF and richness (Holmes & Forsyth, 1995). There are also five types of s-features: length, sentence, retrieval, TF-IDF and richness. All of the s-features are contained in our experiments.
- **H-LSS** (Kim et al., 2015): H-LSS uses the high-dimensional similarity vector compared with LSS. It calculates the elementwise difference between two similarity vectors. Therefore, H-LSS can save more text features than LSS. In (Kim et al., 2015), H-LSS uses

Table 1
Experimental results of different methods for MUIP on the Weibo dataset. At the first line of the table, the numbers represent the count of candidate user pairs. "Pre" represents precision. "Re" represents recall. "F1" represents the F1 score. For the F1 score, the best performance is indicated in bold face.

%		20	40	60	80	100
LSS	Pre	100.00	100.00	100.00	100.00	100.00
	Rec	65.00	67.50	65.00	58.75	58.00
	F1	78.79	80.60	78.79	74.02	73.42
H-LSS	Pre	100.00	100.00	97.22	100.00	98.46
	Rec	50.00	55.00	58.33	60.00	64.00
	F1	66.67	70.97	72.92	75.00	77.58
TF-C	Pre	100.00	100.00	100.00	95.56	96.23
	Rec	50.00	47.50	55.00	53.75	51.00
	F1	66.67	64.41	70.97	68.80	66.67
TF-IDF-C	Pre	100.00	100.00	100.00	100.00	97.87
	Rec	45.00	37.50	46.67	46.25	46.00
	F1	62.07	54.55	63.64	63.25	62.59
DU-DL-I	Pre	93.33	93.33	95.12	94.44	92.75
	Rec	70.00	70.00	65.00	63.75	64.00
	F1	80.00	80.00	77.23	76.12	75.74
DU-DL	Pre	100.00	100.00	93.48	93.44	92.00
	Rec	70.00	75.00	71.67	71.25	69.00
	F1	**82.35**	**85.71**	**81.13**	**80.85**	**78.86**

Table 2
Experimental results of different methods for MUIP on the Douban dataset. At the first line of the table, the numbers represent the count of candidate user pairs. "Pre" represents precision. "Rec" represents recall. "F1" represents the F1 score. For the F1 score, the best performance is indicated in bold face.

%		20	40	60	80	100
LSS	Pre	75.00	73.68	56.25	62.79	62.75
	Rec	30.00	35.00	30.00	33.75	32.00
	F1	42.86	47.46	39.13	43.90	42.38
H-LSS	Pre	71.43	80.00	91.67	86.67	76.19
	Rec	25.00	20.00	18.33	16.25	16.00
	F1	37.04	32.00	30.56	27.37	26.45
TF-C	Pre	100.00	90.00	85.71	61.11	68.42
	Rec	30.00	22.50	20.00	13.75	13.00
	F1	46.15	36.00	32.43	22.45	21.85
TF-IDF-C	Pre	100.00	100.00	92.31	88.24	84.21
	Rec	35.00	25.00	20.00	18.75	16.00
	F1	51.85	40.00	32.88	30.93	26.89
DU-DL-I	Pre	100.00	90.48	92.59	85.00	72.73
	Rec	65.00	47,50	41.67	42.50	40.00
	F1	**78.79**	62.30	57.47	56.67	51.61
DU-DL	Pre	100.00	100.00	94.87	96.00	95.31
	Rec	65.00	65.00	61.67	60.00	61.00
	F1	**78.79**	**78.79**	**74.75**	**73.85**	**74.39**

many text features that uniquely exist in Korean social media. In this paper, the d-features of LSS are adopted in H-LSS.

- **TF-C** (Qian & Liu, 2013): TF can be calculated from the document set of a user. Word unigrams are selected to measure the similarity between two users by calculating the cosine similarity between two TF vectors. This unsupervised method is referred to as TF-Cosine (TF-C).
- **TF-IDF-C** (Qian & Liu, 2013): TF-IDF can also be calculated from the document set of a user. The similarity between two users is measured by calculating the cosine similarity between two TF-IDF vectors. This unsupervised method is referred to as TF-IDF-Cosine (TF-IDF-C).
- **DU-DL-I**: In DU-DL-I, a sentence encoding model is used instead of the sentence pair interaction model. Sentence pair interaction models focus on the relationship between two sentences and aggregates inter-sentence interactions. These models are more effective than sentence encoding models in document similarity calculation. To verify this point, a variant version of DU-DL, DU-DL-I, is selected for comparison. The representative sentence encoding model InferSent (Conneau et al., 2017) is applied to DU-DL-I.

When $\gamma = 0$, DU-DL only uses the keywords of users to calculate user similarity scores, which is similar to the two previously mentioned unsupervised methods. The unsupervised baseline methods only use the cosine function and do not use the deep learning model to calculate user similarity scores.

4.1.3. Performance metrics and parameter settings

The precision, recall and F1-measure are applied to evaluate the proposed algorithm. They are defined as follows:

$$\text{Precision} = \frac{\#\text{correct identified users}}{\#\text{identified users}}, \tag{18}$$

$$\text{Recall} = \frac{\#\text{correct identified users}}{\#\text{identical users}}, \tag{19}$$

$$\text{F1-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision+Recall}}. \tag{20}$$

For DecAtt, batch size is set to 1. The hidden vector size is set to 200. The model is optimized with AdaDelta (Zeiler, 2012) with an initial learning rate $5e^{-4}$. In case of overfitting, dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) is applied before hidden layers with a dropout rate of 0.2.

In DU-DL, the weight γ is set to 0.8 for both datasets. For the $score_d$ calculation strategy, Votes is applied to the Weibo dataset and Sum is applied to the Douban dataset. For the keyword extraction strategy, TextRank is applied to the Weibo dataset and TF-IDF is applied to the Douban dataset. During the training of DecAtt, the ratio of p/t is set to 0.5 on the Weibo dataset and to 0.6 on the Douban dataset. p/t is defined as follows:

$$p/t = \frac{\#\text{positive samples}}{\#\text{total samples}}. \tag{21}$$

The effects of varying the abovementioned parameters and strategies are discussed in detail in Section 4.4.

Ubuntu 16.04 is selected as our experiment platform. GeForce GTX 1080Ti is the graphics card. Python 3.6 and PyTorch 0.4.0 are our main softwares.

4.2. Comparison with baseline methods

This section compares baseline methods with DU-DL on both datasets: Weibo and Douban. The source code of DU-DL is available on request. The experimental results are shown in Tables 1 and 2.

- DU-DL outperforms LSS and H-LSS on both datasets. For two supervised baseline methods, in terms of the F1 score, DU-DL outperforms LSS and H-LSS on the Weibo dataset by 4.66% and 9.15% on average, respectively. On the Douban dataset, DU-DL outperforms two baseline methods by 32.97% and 45.43%, respectively. LSS and H-LSS use manually crafted features to establish models. These manually crafted features consist of lexical features, syntactic features and simple statistical information. The relationships among these features can hardly be captured. However, DU-DL uses the deep learning model to capture the relationship of contextual words. Therefore, DU-DL can capture latent/deep semantic information to learn user writing style. In addition, LSS and H-LSS only capture document-level features, but DU-DL captures both document-level features and user-level features. Therefore, DU-DL is more effective than two supervised methods.
- DU-DL outperforms TF-C and TF-IDF-C on both datasets. For two unsupervised baseline methods, in terms of F1 score, DU-DL outperforms TF-C and TF-IDF-C on the Weibo dataset by 14.28% and 20.56% on average, respectively. On the Douban dataset, DU-DL outperforms two baseline methods by 44.34% and 39.60%, respectively. TF-C and TF-IDF-C use TF and TF-IDF, respectively, to select keywords to calculate the ratio of co-occurring words.

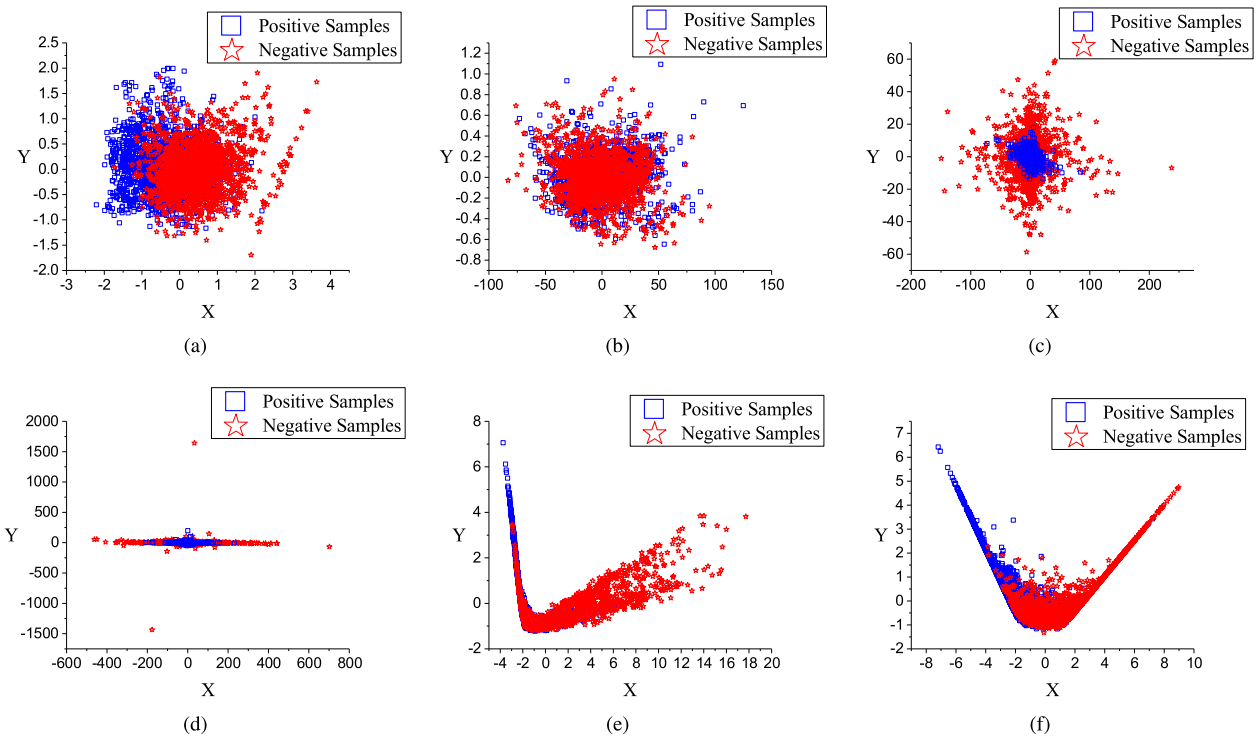


Fig. 4. Visualization of feature vectors on the Weibo dataset. Each node corresponds to a feature vector. "X" and "Y" represent the two principal components of the feature vector. The feature vectors are calculated by two documents or document sets. Positive samples indicate that the two documents or document sets belong to the same author. Negative samples indicate that the two documents or document sets belong to different authors. (a) LSS. (b) H-LSS. (c) TF-C. (d) TF-IDF-C. (e) DU-DL-I. (f) DU-DL.

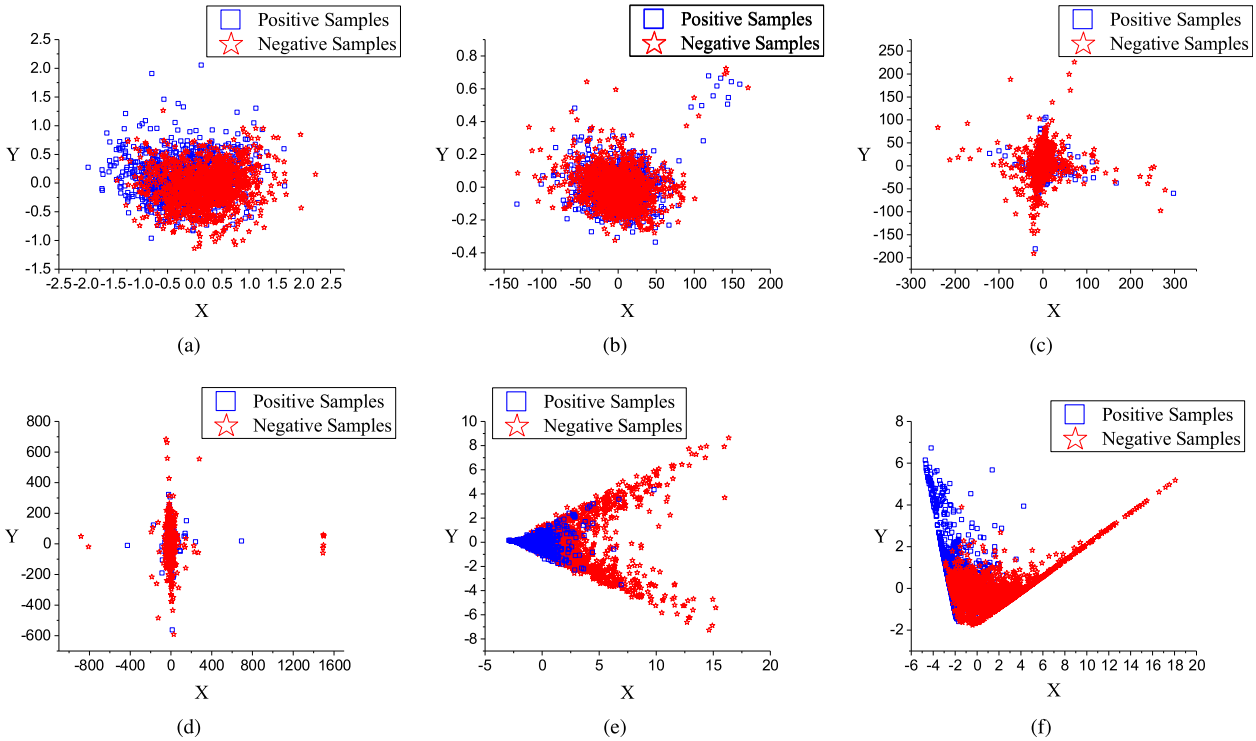


Fig. 5. Visualization of feature vectors on the Douban dataset. Each node corresponds to a feature vector. "X" and "Y" represent the two principal components of the feature vector. The feature vectors are calculated by two documents or document sets. Positive samples indicate that the two documents or document sets belong to the same author. Negative samples indicate that the two documents or document sets belong to different authors. (a) LSS. (b) H-LSS. (c) TF-C. (d) TF-IDF-C. (e) DU-DL-I. (f) DU-DL.

These words are considered simple symbols in this situation. There are no relationships among these symbols. In addition, TF-C and TF-IDF-C capture only user-level features. Therefore, DU-DL is more effective than two unsupervised methods.

- DU-DL achieves the better performance than DU-DL-I. The experimental results show that DU-DL outperforms DU-DL-I by 3.96% and 14.75% on average on the Weibo dataset and Douban dataset, respectively. DecAtt and InferSent are utilized in DU-DL and DU-DL-I, respectively. DecAtt pays attention to the relationship between two documents, while InferSent is skilled in the vector representation of a document. Therefore, DecAtt is more effective than InferSent in document similarity calculation.

4.3. Deep semantic features vs. Manually crafted features

According to the comparison with baseline methods, DU-DL outperforms all the baseline methods. DU-DL-I achieves similar or even superior results compared with other baseline methods. The reason is that they extract different levels of features for MUIP. DU-DL and DU-DL-I use deep learning models to extract deep semantic features. Other baseline methods use traditional feature engineering methods to extract manually crafted features. To determine the distinction between deep semantic features and manually crafted features, PCA is applied in this paper as discussed in (Wu et al., 2017).

PCA is a type of dimensionality reduction method. It can map original high-dimensional feature vectors into a low-dimensional space. Thus, the distinction of these feature vectors can be determined easily. Specifically, this paper first selects feature vectors from the feature extraction process of each method. Then, PCA maps these feature vectors into two-dimensional (2-D) space. Finally, the vectors in 2-D space can be visualized easily. The details of the process are given as follows.

Firstly, the feature vectors should be selected in the feature extraction process of each method. LSS and H-LSS select their similarity vectors. TF-C calculates the elementwise difference between two TF vectors. TF-IDF-C calculates the elementwise difference between two TF-IDF vectors. DU-DL-I selects the vectors that are output by the last hidden layer of InferSent. DU-DL selects the vectors that are output by the last hidden layer of DecAtt.

Then, PCA is applied to map these feature vectors into 2-D space. PCA is selected in this paper because it can show the linear distribution of feature vectors in the proposed method. For each method, PCA is performed as follows:

(1) This step calculates the covariance matrix $\boldsymbol{A}$ of the feature matrix $\boldsymbol{F}$. The feature matrix $\boldsymbol{F}$ consists of all the feature vectors. The covariance matrix $\boldsymbol{A}$ measures how much the features vary from the mean. The covariance matrix $\boldsymbol{A}$ is calculated as follows:

$$\boldsymbol{A} = \frac{1}{m-1}\sum_{i=1}^{m}(\boldsymbol{F}_i - \overline{\boldsymbol{F}})(\boldsymbol{F}_i - \overline{\boldsymbol{F}})^T, \tag{22}$$

where m is the number of input feature vectors; $\boldsymbol{F}_i$ is the ith feature vector; $\overline{\boldsymbol{F}}$ is the mean of all the feature vectors; $\boldsymbol{F} \in \mathbb{R}^{m\times d}$; $\boldsymbol{A} \in \mathbb{R}^{d\times d}$; and d is the dimension of the feature vectors.

(2) This step performs a dimensionality reduction operation. The eigenvectors and eigenvalues of the covariance matrix $\boldsymbol{A}$ should be calculated. The first and second eigenvectors are selected with the sorted eigenvalues in descending order. The two eigenvectors represent the dominant principal components of the feature matrix $\boldsymbol{F}$. A mapping matrix $\boldsymbol{E}_{1,2} \in \mathbb{R}^{2\times d}$ can be constructed by the two eigenvectors. The dimensionality reduction is performed as follows:

$$\boldsymbol{D} = \boldsymbol{E}_{1,2}\boldsymbol{F}^T, \tag{23}$$

where $\boldsymbol{D} \in \mathbb{R}^{2\times m}$ is the matrix after dimensionality reduction.

Finally, the vectors in 2-D space can be visualized. The visualization results with different methods are shown in Figs. 4 and 5. The nodes that belong to positive samples are denoted as square shapes. The nodes that belong to negative samples are denoted as star shapes. A good visualization performance is that the nodes of the same shape should be clustered and should not overlap with the nodes of other shapes.

From Figs. 4 and 5, several observations can be found as follows:

Table 3
Effectiveness of document-level feature extraction and user-level feature extraction. "User-Level" represents that DU-DL only uses the user-level feature extraction method. "Document-Level" represents that DU-DL only uses the document-level feature extraction method. The results are evaluated on average by the F1 score.

	Weibo	Douban
User-Level	55.49	40.91
Document-Level	**85.25**	**74.00**

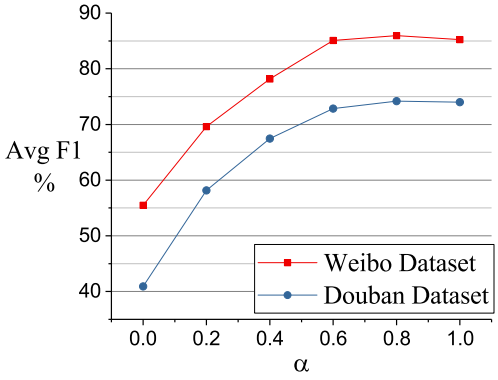


Fig. 6. Impact of parameter γ. γ is the weight. The larger the γ is, the larger is the weight that can be assigned on document-level feature extraction.

- For vertical comparison, DU-DL achieves the best performance in all the methods on the same dataset. The nodes of baseline methods with different shapes heavily overlap. However, there are comparatively distinct boundaries of the nodes with different shapes in DU-DL. It demonstrates that the feature extraction process of DU-DL is more effective than baseline methods. The reason is that DU-DL extracts deep semantic features by DecAtt. In addition, DU-DL uses DecAtt to aggregate interdocument interactions. For LSS, H-LSS, TF-C and TF-IDF-C, they extract only manually crafted features. Although DU-DL-I extracts deep semantic features by InferSent, it focuses on the vector representation of the document but does not exploits the interdocument interactions.
- For horizontal comparison, the nodes of DU-DL with the same shape show regular linear distributions on both datasets. The nodes of DU-DL-I are clustered at a narrowed angle and lack dispersion. The distributions of the nodes of other baseline methods cannot be distinguished. It can be inferred that deep semantic features can show more similar characteristics than manually crafted features on different datasets.

According to the previous analysis, deep semantic features that belong to different categories of samples can be better determined than manually crafted features. It demonstrates that deep semantic features are more effective than manually crafted features in expressing user writing styles.

4.4. Discussions

4.4.1. Document-level feature extraction vs. User-level feature extraction

In DU-DL, the similarity score of two different users is calculated by Eq. (17), where γ is the weight. The larger γ is, the larger is the weight that can be assigned to document-level feature extraction. When $\gamma = 1.0$, DU-DL extracts only document-level features to learn the user writing style. When $\gamma = 0.0$, DU-DL extracts only user-level features. To compare the effectiveness of document-level feature extraction with user-level feature extraction, γ is set to 1.0 and 0.0, respectively. The experimental results are shown in Table 3. In addition, to verify the performance of DU-DL with different values of γ, DU-DL sets γ =

0.0, 0.2, 0.4, 0.6, 0.8, 1.0. The experimental results are shown in Fig. 6. Several observations can be found as follows:

- For two datasets, the experimental results demonstrate that document-level feature extraction is more effective than user-level feature extraction in expressing user writing style. With $\gamma = 1.0$, the result increases 29.76% compared with $\gamma = 0.0$ on the Weibo dataset. On the Douban dataset, the result increases 33.09%. The reason is that document-level feature extraction expresses text features from each single document, but user-level feature extraction only extracts the keywords of the document set of a user.
- Document-level feature extraction has a more important role in DU-DL. On both datasets, it can be observed that the performance of DU-DL initially rises rapidly with an increase in the value of γ. When γ is larger than 0.8, the performance decreases slowly. An excessively large γ may enforce DU-DL to disregard user-level features of texts. On the other hand, an excessively small γ may disregard document-level features. Therefore, the overall performance can be increased when more weight is assigned to document-level feature extraction.
- User-level feature extraction also has an important role in DU-DL. On both datasets, it can be observed that when γ is larger than 0.8, the performance decreases slowly. It demonstrates that user-level feature extraction is effective when it is assigned a lower weight. DU-DL uses manually crafted feature extraction methods to extract keywords in user-level feature extraction. These methods pay attention to the important parts of these sentences before extracting deep semantic features. However, other deep learning methods can only directly extract deep semantic features from original sentences. Therefore, DU-DL uses manually crafted feature extraction methods to reprocess the sentences. Based on this finding, the deep learning model can more effectively extract deep semantic features.

4.4.2. Effects of different $score_d$ calculation strategies

Votes and Sum are two $score_d$ calculation strategies in DU-DL. Votes pays attention to the positive classification results, but Sum pays attention to all the classification results. To investigate the sensitivity of our choice of Votes and Sum, DU-DL uses Votes and Sum to calculate $score_d$, respectively. Other experimental parameters are fixed. The experimental results are shown in Fig. 7.

The experimental results demonstrate that the Votes strategy is effective on the Weibo dataset and the Sum strategy is effective on the Douban dataset. By using the Votes strategy, the results increase 4.61% on average compared with using the Sum strategy on the Weibo dataset. On the Douban dataset, the results decrease by 1.65% on average. A possible reason is that the average document length of the Douban dataset is longer than that of the Weibo dataset. The longer the text lengths of two documents, the more information can be obtained to judge the relationship between the two documents. It can be inferred that the similarity score of two documents is more accurate in long texts than in short texts. The Sum strategy is calculated by adding the similarity scores. Therefore, the Sum strategy is more effective on long text datasets than on short text datasets.

4.4.3. Effects of different keyword extraction strategies

TF, TF-IDF and TextRank are three keyword extraction strategies. They are used to extract keywords from the document set of a user. To verify the effectiveness of DU-DL by using different keyword extraction strategies, DU-DL uses TF, TF-IDF and TextRank to extract keywords. Other experimental parameters are fixed. The experimental results are shown in Fig. 8.

The experimental results demonstrate that TextRank is the most effective strategy on the Weibo dataset and that TF-IDF is the most effective strategy on the Douban dataset. On the Weibo dataset, the results using TextRank increase 3.16% and 4.63% on average compared with using TF and TF-IDF, respectively. On the Douban dataset, the results using TF-IDF increase 6.70% and 7.41% on average compared with using TF and TextRank, respectively. It can be inferred that TextRank is effective in extracting keywords on short text datasets. It is easy to judge the relationships among contextual words because of the short distance between two keywords in a short text. In contrast, TextRank is less effective than TF-IDF on long text datasets. Therefore, TextRank is employed on the Weibo dataset and TF-IDF is utilized on the Douban dataset.

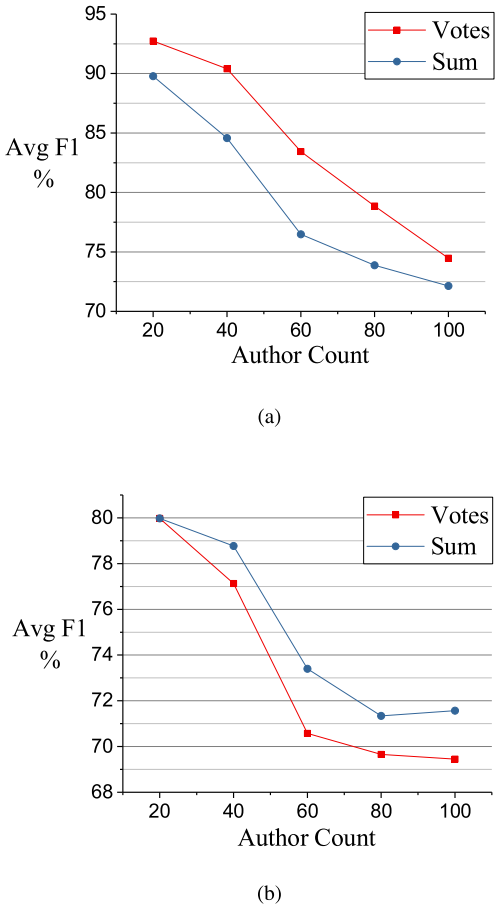
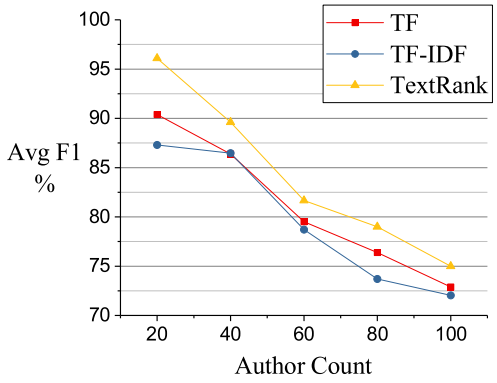


Fig. 7. Effectiveness of different $score_d$ calculation strategies: Votes and Sum. (a) Weibo. (b) Douban.

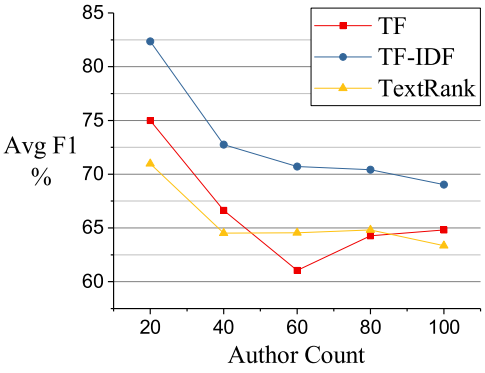
4.4.4. Effects of different ratio of p/t

During the training of DecAtt, p/t needs to be set properly in case of class imbalance. To investigate the sensitivity of p/t, the experiments set $p/t = 0.3, 0.4, 0.5, 0.6, 0.7$. The experimental results are shown in Fig. 9.

DU-DL achieves the best performance by setting p/t to 0.5 and 0.6 on the Weibo dataset and Douban dataset, respectively. In terms of the F1 score, DU-DL achieves the best result of 85.95% with $p/t = 0.5$ on the Weibo dataset. On the Douban dataset, DU-DL achieves the best result of 76.11% with $p/t = 0.6$. It can be observed that the performances decrease rapidly when $p/t > 0.6$ on both datasets. It demonstrates that the training sets may be imbalanced when $p/t = 0.7$. When p/t is set between 0.3 and 0.6, DU-DL achieves the best performance with $p/t = 0.5$ on the Weibo dataset. However, DU-DL achieves the worst performance with $p/t = 0.5$ on the Douban dataset. When $p/t = 0.5$, the number of positive samples is equal to the number of negative samples. The training sets are least affected by class imbalance. Therefore, it can be inferred that the training samples that belong to different categories

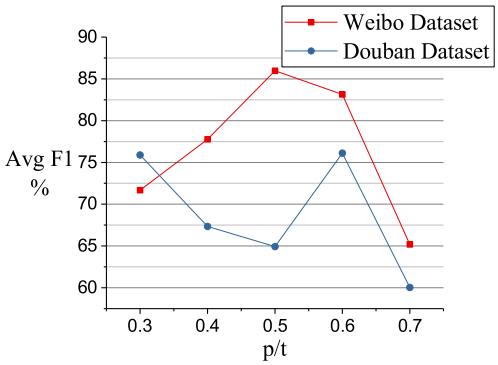


(a)



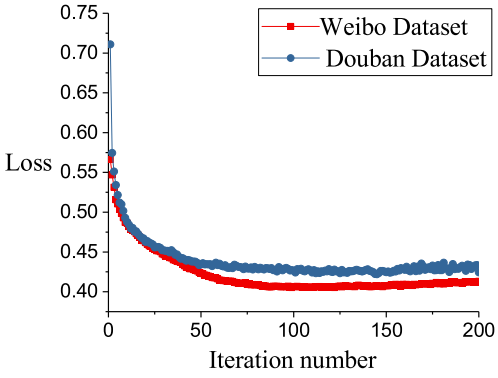
(b)

Fig. 8. Effectiveness of different keyword extraction strategies: TF, TF-IDF and TextRank. (a) Weibo. (b) Douban.



(a)

Fig. 9. Impact of parameter p/t.



(a)

Fig. 10. Convergence curves of DecAtt in DU-DL.

are better distinguished on the Weibo dataset than on the Douban dataset.

4.4.5. Convergence analysis

To investigate the convergence speed of the deep learning model, DecAtt, the convergence curves are plotted to illustrate the variation in loss with an increase in iteration number. The loss is calculated as the difference between the real value and the predicted score across the entire testing set. The convergence curves are shown in Fig. 10.

It can be observed that DecAtt is able to converge in fewer than 100 iterations on the Weibo dataset. On the Douban dataset, DecAtt is able to converge in less than 90 iterations. There are no noticeable decrements after these iterations. In short, DecAtt can converge in a limited number of iterations.

5. Conclusion

This paper proposes a method, learning document-level and user-level features with deep learning (DU-DL) to solve multiple userid identification problem (MUIP). DU-DL has two stages: deep semantic feature extraction and candidate user identification. In the deep semantic feature extraction stage, DU-DL extracts deep semantic features from the document level and user level. In the candidate user identification stage, deep semantic features of different levels are combined to identify the most similar user pairs.

In the future, DU-DL can be extended in several ways as follows. Firstly, some state-of-the-art deep learning models such as BERT (Devlin et al., 2019), siamese network (Ji, Zhang, Jie, Ma, & Jonathan Wu, 2021) and other models for text matching (Liu, Zhang, Xu, & Chen, 2021; Zhao, Lu, Li, Yu, Jian et al., 2021), can be employed to solve MUIP. Secondly, the deep semantic features and manually crafted features can be better fused, such as concatenating them to construct the final feature space (Cao, Wang, Li, Wang, Ding et al., 2020; Madisetty & Desarkar, 2018; Majumder, Poria, Gelbukh, & Cambria, 2017; Xue et al., 2018). Finally, DU-DL can be extended to solve user identity linkages in multiple different social networks (Liu, Shen, Guan, & Zhou, 2020; Shu, Wang, Tang, Zafarani, & Liu, 2017).

CRediT authorship contribution statement

Xin Du: Conceptualization, Methodology, Software, Investigation, Data curation, Writing – original draft. **Siyuan Chen:** Writing – review & editing. **Zhiyue Liu:** Writing – review & editing. **Jiahai Wang:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the National Key R&D Program of China (2018AAA0101203), the National Natural Science Foundation of China (62072483), and the Guangdong Basic and Applied Basic Research Foundation (2022A1515011690, 2021A1515012298).

References

Afroz, S., Islam, A. C., Stolerman, A., Greenstadt, R., & Mccoy, D. (2014). Doppelgänger finder: Taking stylometry to the underground. In *2014 IEEE symposium on security and privacy* (pp. 212–226).

Alharthi, R., Alhothali, A., & Moria, K. (2021). A real-time deep-learning approach for filtering Arabic low-quality content and accounts on Twitter. *Information Systems*, *99*, Article 101740.

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 632–642).

Cao, N., Ji, S., Chiu, D. K., & Gong, M. (2022). A deceptive reviews detection model: Separated training of multi-feature learning and classification. *Expert Systems with Applications*, *187*, Article 115977.

Cao, J., Wang, S., Li, B., Wang, X., Ding, Z., & Wang, F.-Y. (2020). Integrating multisourced texts in online business intelligence systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *50*(5), 1638–1648.

Che, W., Li, Z., & Liu, T. (2010). LTP: A Chinese language technology platform. *Journal of Chinese Information Processing*, *2*(6), 13–16.

Chen, H. C., Goldberg, M., & Magdonismail, M. (2004). Identifying multi-ID users in open forums. In *International conference on intelligence and security informatics* (pp. 176–186). Berlin, Heidelberg: Springer.

Chen, X., Li, M., Gao, S., Chan, Z., Zhao, D., Gao, X., Zhang, X., & Yan, R. (2022). Follow the timeline! Generating abstractive and extractive timeline summary in chronological order. *ACM Transactions on Information Systems*, In Press.

Chen, Q., Zhu, X., Ling, Z., Wei, S., Jiang, H., & Inkpen, D. (2016). Enhanced LSTM for natural language inference. In *Proceedings of the 55th annual meeting of the association for computational linguistics* (pp. 1657–1668).

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 670–680).

De Boer, P., Kroese, D. P., Mannor, S., & Rubinstein, R. Y. (2005). A tutorial on the cross-entropy method. *Annals of Operations Research*, *134*(1), 19–67.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies* (pp. 4171–4186).

Fahfouh, A., Riffi, J., Adnane Mahraz, M., Yahyaouy, A., & Tairi, H. (2020). PV-DAE: A hybrid model for deceptive opinion spam based on neural network architectures. *Expert Systems with Applications*, *157*, Article 113517.

He, H., & Lin, J. (2016). Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies* (pp. 937–948).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Holmes, D. I., & Forsyth, R. S. (1995). The federalist revisited: New directions in authorship attribution. *Literary and Linguistic Computing*, *10*(2), 111–127.

Hosseinia, M., & Mukherjee, A. (2017). Detecting sockpuppets in deceptive opinion spam. In *International conference on computational linguistics and intelligent text processing* (pp. 255–272).

Hu, Y., Wang, S., Ren, Y., & Choo, K.-K. R. (2018). User influence analysis for Github developer social networks. *Expert Systems with Applications*, *108*, 108–118.

Jain, L., Katarya, R., & Sachdeva, S. (2020). Opinion leader detection using whale optimization algorithm in online social network. *Expert Systems with Applications*, *142*, Article 113016.

Ji, Y., Zhang, H., Jie, Z., Ma, L., & Jonathan Wu, Q. M. (2021). CASNet: A cross-attention siamese network for video salient object detection. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(6), 2676–2690.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*(1), 11–21.

Khoory, S., Al Abdooli, M., Al Roken, N., & Hacid, H. (2019). Speculo: A tool for multiple identities exploration and detection in social networks. In *IEEE/WIC/ACM international conference on web intelligence (vol. 24800)* (pp. 297–300).

Kim, K., Noh, Y., & Park, S.-B. (2015). Detecting multiple userids on Korean social media for mining TV audience response. In *IEEE region 10 annual international conference* (pp. 1–4).

Kumar, S., Cheng, J., Leskovec, J., & Subrahmanian, V. (2017). An army of me: Sockpuppets in online discussion communities. In *Proceedings of the 26th international conference on world wide web* (pp. 857–866).

Lan, W., & Xu, W. (2018). Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. In *Proceedings of the 27th international conference on computational linguistics* (pp. 3890–3902).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Li, H. (2018). Deep learning for natural language processing: Advantages and challenges. *National Science Review*, *5*(1), 24–26.

Li, Z., Zhang, M., Che, W., Liu, T., Chen, W., & Li, H. (2011). Joint models for Chinese POS tagging and dependency parsing. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1180–1191).

Liu, X., Shen, C., Guan, X., & Zhou, Y. (2020). We know who you are: Discovering similar groups across multiple social networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *50*(7), 2693–2704.

Liu, M., Zhang, Y., Xu, J., & Chen, Y. (2021). Deep bi-directional interaction network for sentence matching. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, *51*(7), 4305–4329.

Madisetty, S., & Desarkar, M. S. (2018). A neural network-based ensemble approach for spam detection in Twitter. *IEEE Transactions on Computational Social Systems*, *5*(4), 973–984.

Majumder, N., Poria, S., Gelbukh, A., & Cambria, E. (2017). Deep learning-based document modeling for personality detection from text. *IEEE Intelligent Systems*, *32*(2), 74–79.

Manaskasemsak, B., Tantisuwankul, J., & Rungsawang, A. (2021). Fake review and reviewer detection through behavioral graph partitioning integrating deep neural network. *Neural Computing and Applications*.

Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing order into texts. In *Proceedings of the 2004 conference on empirical methods in natural language processing* (pp. 404–411).

Ming, Y., Cao, S., Zhang, R., Li, Z., Chen, Y., Song, Y., & Qu, H. (2017). Understanding hidden memories of recurrent neural networks. In *2017 IEEE conference on visual analytics science and technology* (pp. 13–24).

Mohammadmosaferi, K. K., & Naderi, H. (2020). Evolution of communities in dynamic social networks: An efficient map-based approach. *Expert Systems with Applications*, *147*, Article 113221.

Nie, Y., & Bansal, M. (2017). Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd workshop on evaluating vector space representations for NLP* (pp. 41–45).

Novak, J., Raghavan, P., & Tomkins, A. (2004). Anti-aliasing on the web. In *International conference on world wide web* (pp. 30–39).

Page, L. (1998). The PageRank citation ranking: Bringing order to the web. *Stanford Digital Libraries Working Paper*, *9*(1), 1–14.

Parikh, A. P., Täckström, O., Das, D., & Uszkoreit, J. (2016). A decomposable attention model for natural language inference. In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 2249–2255).

Qian, T., & Liu, B. (2013). Identifying multiple userids of the same author. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1124–1135).

Ren, Z., Shen, Q., Diao, X., & Xu, H. (2021). A sentiment-aware deep learning approach for personality detection from text. *Information Processing and Management*, *58*(3), Article 102532.

Rouast, P. V., Adam, M., & Chiong, R. (2021). Deep learning for human affect recognition: Insights and new developments. *IEEE Transactions on Affective Computing*, *12*(2), 524–543.

Shen, D., Wang, G., Wang, W., Min, M. R., Su, Q., Zhang, Y., Li, C., Henao, R., & Carin, L. (2018). Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *56th annual meeting of the association for computational linguistics, proceedings of the conference* (pp. 440–450).

Shu, K., Wang, S., Tang, J., Zafarani, R., & Liu, H. (2017). User identity linkage across online social networks: A review. *ACM SIGKDD Explor. Newsl.*, *18*(2), 5–17.

Solorio, T., Hasan, R., & Mizan, M. (2013). A case study of sockpuppet detection in wikipedia. In *Proceedings of the workshop on language analysis in social media* (pp. 59–68).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. In *Advances in neural information processing systems* (pp. 2692–2700).

Wang, L., Che, W., & Liu, T. (2009). An SVMTool-based Chinese POS tagger. *Journal of Chinese Information Processing*, *23*(4), 16–22.

Wang, H., Lu, Z., Li, H., & Chen, E. (2013). A dataset for research on short-text conversation. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 935–945).

Wu, T., Sheng, W., Liu, S., Zhang, J., Yang, X., Alrubaian, M., & Hassan, M. M. (2017). Detecting spamming activities in twitter based on deep-learning technique. *Concurrency and Computation Practice and Experience*, *29*(19), Article e4209.

Xue, D., Wu, L., Hong, Z., Guo, S., Gao, L., Wu, Z., Zhong, X., & Sun, J. (2018). Deep learning-based personality recognition from text posts of online social networks. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, *48*(11), 4232–4246.

Yamak, Z. R. (2018). *Multiple identities detection in online social media* (Ph.D. thesis).

Yang, F., Quan, X., Yang, Y., & Yu, J. (2021). Multi-document transformer for personality detection. In *Proceedings of the AAAI conference on artificial intelligence (vol. 35)* (pp. 14221–14229).

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, *13*(3), 55–75.

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. arXiv:1212.5701.

Zhang, M., Deng, Z., Che, W., & Liu, T. (2012). Combining statistical model and dictionary for domain adaption of Chinese word segmentation. *Journal of Chinese Information Processing*, *26*(2), 8–12.

Zhao, P., Lu, W., Li, Y., Yu, J., Jian, P., & Zhang, X. (2021). Chinese semantic matching with multi-granularity alignment and feature fusion. In *2021 International joint conference on neural networks*.