Expert Systems With Applications 195 (2022) 116600

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A network-based feature extraction model for imbalanced text data

Keping Li[a], Dongyang Yan[a,*,1], Yanyan Liu[a], Qiaozhen Zhu[b]

[a] *State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing 100044, China*
[b] *School of Traffic and Transportation, Beijing Jiaotong University, Beijing 100044, China*

A R T I C L E I N F O

Keywords:
Complex Network
CNN
Text Analysis
Imbalanced Data
Random Walk

A B S T R A C T

The explosive growth of text data has attracted many researchers to explore the efficient method to extract valuable hidden information. Many technologies, especially deep learning methods, have achieved great success in text analysis. However, the most powerful methods always require a considerable quantity of data for training, which may suffer from imbalanced data in some cases. In this paper, we propose a network-based Convolution Neural Network (NCNN) to mitigate the effect of imbalanced data. The proposed model first generates new synthetic samples for the imbalanced data based on the random walking of the network. Then an extra layer called Polar Layer is introduced to connect the output from the network model of the text to the classical CNN. Two electing strategies (n-NCNN and x-NCNN) are proposed to improve the performance of NCNN further. In the experimental section, the proposed model is applied to *Reuters 21578* and *WebKb*. By comparing with six approaches, we prove the effectiveness of the proposed NCNN model on the imbalanced text data.

1. Introduction

The rapidly developing internet technology comes with the appearance of a large amount of text data every day, making text analysis a promising field for researchers. For text analysis, feature extraction is the basis for extracting the primary information to represent the original text. The main object of feature extraction is to reduce the dimension of the feature representation and eliminate the noise in the feature space. The features of text data refer to the language units (words) or phrases (Collobert et al., 2011), and even characters, which also perform good results in specific tasks (Zhang et al., 2015). There are numerous machine learning methods to achieve feature extraction, and their common advantage is that they are well adaptive to different tasks (Hassan et al., 2015; Junejo et al., 2016; Prihatini et al., 2018; Zhao, & Mao, 2018; Gupta, & Gupta, 2021; Yan et al., 2020). In recent years, the success of distributed representation of words (Mikolov et al., 2013; Le, & Mikolov, 2014; Pennington et al., 2014; Devlin et al., 2019) has inspired researchers to learn the features of text through training on the neural network (Kim, 2014; Hu et al., 2014; Yin et al., 2016; Foland, & Martin, 2017, Liang et al, 2017; Young et al., 2018). At the same time, the explosion of data on the internet provides large datasets for deep learning methods. Based on a large amount of training data, the pre-training and transformer have achieved a significant improvement on Natural Language Processing (NLP) (Vaswani et al., 2017; Howard, & Ruder, 2018; Sun et al., 2020).

The excellent performance of most of the methods is based on considerately selected balanced data. However, the practical scenarios of text data analysis always come with the imbalanced distribution of different labels. For example, the comments on specific affairs may contain a majority of positive opinions and a minority of negative ones. The small number of negative comments may be of great value to helping the decision-makers make the right decisions. Moreover, the wrong decision making may lead to overwhelmingly serious consequences in imbalanced text data analysis. A cancer patient being misdiagnosed as healthy is far worse than the consequences of a healthy person being diagnosed with cancer in the medical field.

The imbalanced data always contain the part of majority classes that have many samples and the part of minority classes, which are represented by a much smaller number of samples than the majority classes. Because most of the models are constructed assuming that the training data is balanced, the performance is reduced on the imbalanced data. The performance reduction always happens in the minority classes as the models tend to neglect the samples of minority classes to get high accuracy of the majority classes (He, & Garcia, 2009). Though numerous researches are focusing on imbalanced numerical data (He, & Garcia, 2009; Krawczyk, 2016; Gao et al., 2017), few studies claim that their

* Corresponding author.
E-mail addresses: kpli@bjtu.edu.cn (K. Li), dongyangyan@bjtu.edu.cn (D. Yan), 19114006@bjtu.edu.cn (Y. Liu), 17114232@bjtu.edu.cn (Q. Zhu).
[1] ORCID: 0000-0001-6598-7705.

https://doi.org/10.1016/j.eswa.2022.116600
Received 5 September 2020; Received in revised form 18 January 2022; Accepted 20 January 2022
Available online 2 February 2022

model can be applied to text data analysis. Some effective methods for analyzing numerical data could not be directly used to text data because the information of text is contained in word sequences rather than numbers. Moreover, as far as we know, most of the methods for processing imbalanced text data must first convert a document to numerical data (Ogura, et al., 2011; Iglesias et al., 2013; Naderalvojoud et al., 2015; Song et al., 2016; Li et al., 2018; Xiao et al., 2019), i.e., the Bag-of-Words (BoW) vectors (Iglesias et al., 2013; Naderalvojoud et al., 2015; Song et al., 2016). This two-step solution may lead to two kinds of information losses: the loss of conversion from text to vectors and the loss of sampling (when used).

To this end, we propose a simple yet useful feature extraction model to over-sample and expand the number of samples to deal with the problem of imbalanced text data from the angle of text data rather than numerical data. The proposed model uses a powerful network tool to generate synthetic samples through random walking. We contend that the random walk paths, which have been successfully used for graph embedding (Perozzi et al., 2014; Grover, & Leskovec, 2016; Hamilton et al., 2017), will reconstruct the structural and statistical properties of a text. The main idea is to use the random walk paths of a document to represent its primary information. Though the random walk paths, which are used to capture the continuous feature representation of nodes in the network (Grover, & Leskovec, 2016), lose the word order in the documents, they keep the high-level structural and semantic information. The over-sampled documents within the same class contain specific high-level information while behaving in absolutely different word sequences, alleviating the effect of overfitting during training. We also introduce a new neural network layer to transfer the random paths to a form that is appropriate for a CNN model (we call their assembling as NCNN) to make the model adapt to the existing neural network framework.

The main contributions of this paper are summarized:

1. A network-based feature extraction model is proposed for processing imbalanced text data. As far as we know, we are the first to introduce a random walk strategy to deal with text data imbalance.
2. We explore the methods to combine random walk paths with CNN and propose the Polar Layer as their intermedia.
3. We introduce an electing strategy to improve the performance of NCNN further. In the experimental section, the NCNN model with an electing strategy achieves the best results.
4. By comparing with different methods of imbalanced text data, we certify the performance of the proposed model.

We organize the rest of this paper as follows. The next section reviews related works, including methods on imbalanced text data, complex networks, and random walks. In section 3, we introduce the proposed model and present the framework of NCNN. Section 4 shows the experimental results to verify the performance of our model. In section 5, we present the conclusions of this paper.

2. Related works

2.1. Methods on imbalanced text data

Over the last decades, handling data imbalance is always the focus of industry and academia. The methods to deal with such issues can be categorized into data sampling, algorithm modification, and cost-sensitive learning (He & Garcia, 2009; Krawczyk, 2016; Li et al., 2018). As for the text data, most of the studies concentrate on data sampling and algorithm modification. So we mainly review these two types of methods. To make them convenient for the review, we further divide these approaches into three groups: direct-sampling methods, term-weighting methods, and background-knowledge-based methods.

The direct-sampling methods aim to re-sample the data through recombination of the original data to produce new samples of minority classes or reduce existing samples of majority classes. A simple way of direct sampling is to copy or remove the randomly chosen samples from the original datasets to equalize the class membership. However, this approach only balances the number of samples between classes without changing the issues of feature imbalance, which can only get very limited improvement. SMOTE (Chawla et al., 2002) is first proposed for numerical data and can be used in text feature vectors (e.g., BoW) (Ma et al., 2018). The idea of SMOTE is to get a synthetic vector from two neighbors of the minority class. The digit value in each dimension of the synthetic vector is the weighted summation (the weight is generated randomly) of the two neighbors. Because a random process is introduced, the new samples from SMOTE are forced to be unspecific to lead a general decision region, which is useful to avoid overfitting. An HMM model is introduced in the literature (Iglesias et al., 2013) to generate feature words and their weight to get new synthetic samples, which helps get better results than SMOTE in dealing with some imbalanced text datasets. Wang et al. (2013) present an algorithm of boundary region cutting (BRC) to alleviate boundary ambiguity of two-class text classification. BRC is an under-sampling operation by which the samples in the dense boundary region of the majority class are randomly eliminated. *Li et al.* find that the random elimination in BRC can not effectively remove the samples in the majority class near minority class samples. So they propose a local dense mixed region cutting (LDMRC) algorithm (Li et al., 2019) to improve BRC. An ensemble method is proposed by Zuo et al. (2016) to improve the latent Dirichlet allocation (LDA) topic model on imbalanced short text data. The short texts in the same class are collected as a whole to construct a co-occurrence complex network. Then pseudo-document sets are sampled from the neighbors of every node in the network. Finally, the LDA model is trained based on the pseudo-document sets. Song et al. (2016) explore a combination of over-sampling and under-sampling based on the K-means clustering. The number of samples in the majority and minority classes is supposed to reach a certain value k. For the minority classes, the samples are clustered into two groups, and then a new sample is generated by SMOTE for the smaller group. This process is repeated until the number of samples reaches k. On the flip side, the samples are clustered into k groups for the majority classes, and k samples nearest to the cluster centers are left, yielding the balanced majority classes.

Term-weighting methods give feature words a specific weight to balance the role of each feature word. In literature (Ogura et al., 2011), the authors compare the influence of different metrics for feature selection in the imbalanced text data. The authors introduce three types of metrics, including six different metrics: χ_p^2 and Gini index for Type-1; χ^2 and information gain for Type-II; and signed χ^2 and signed information gain for Type-III. The results show that Type-1 and Type-II achieve comparable performance (better than Type-II). And based on the results, they conclude that negative features are beneficial in classifying imbalanced text data. It should be noted that being positive or negative for the features refers to their contribution to a certain category, e.g., a positive (negative) feature for class i may specifically occur more (less) frequently in this class. TFIDF is a widely used weighting scheme because it is a more effective way than term frequency (TF) to represent the relative weight to the overall condition of the whole datasets by introducing inverse document frequency (IDF) (Ogura et al., 2011). However, TFIDF neglects the in-class information of the datasets and thus can not consider class-specific term weight. *Naderalvojoud et al.* (Naderalvojoud et al., 2015) propose a positive and negative-based term weighting scheme to take the category membership into account. This idea can mitigate the effects of unbalanced data better than TFIDF because the proposed term weighting scheme associates terms with each prior category information. Wu et al. (2014) present ForesTexter to handle the problem of text data imbalance in binary classification using a random forest algorithm. ForeTexter uses a support vector machine (SVM) classifier to classify features into two groups to split every branch successively. By splitting the samples into feature subspaces and retaining both negative and positive features in every subspace, this algorithm decreases the influence of data imbalance.

There are also background-knowledge-based methods that use domain-specific knowledge as supplementary information to improve the performance of domain-sensitive tasks on imbalanced text data. The idea of these methods is also to implement over-sample to balance the datasets like direct-sampling methods, while the difference occurs in the use of background knowledge. Literature (Li et al., 2018) considers the sentimental polarity of text sentiment classification and introduces "inversion" and "imitation" strategies to generate new samples. The "inversion" strategy converts a document in majority class into the text with opposite sentimental polarity, e.g., "I love you" to "I hate you." On the contrary, the "imitation" strategy changes the sentimental words into their synonymous words, e.g., "I love you" to "I like you." In recent years, deep learning methods are also introduced to tackle the issues of processing imbalanced text data. Xiao et al. (2019) use transfer learning to first train a model in the balanced data and then apply the model to the imbalanced data with fine-tuning. Because the pre-trained model contains information beyond the text datasets, it can achieve more stable performance than the model trained directly on the text datasets. New powerful generation models like GANs, BERT, and GPT-2 are pre-trained from a huge amount of text and thus are able to produce fluent pseudo text. Some researchers concentrate on generating new text samples directly through text generation model. One strategy is to generate new text through the generation model after a fine-tuning based on the imbalanced datasets (Shaikh et al., 2021). Tang et al. (2021) present another feasible strategy: first, translate a document into another language and then translate it back. The documents translated back have the same topics but different expressions. However, these two strategies both have their limitations. For the first strategy, fine-tuning, which still needs enough text samples as the training data, is essential for the final production. Meanwhile, the pre-trained models may highly extend the boundary of the target datasets, which introduces extra noise to the minority classes that don't contain enough samples for fine-tuning. For the second strategy, the translation of a document only increases the description styles instead of the new samples with the same topic. The authors introduce another operation like "imitation" mentioned above to enrich samples.

Above all, most direct-sampling methods and term-weighting methods have to deal with the problem of imbalance at the numerical level instead of the text level. The background-knowledge-based methods achieve the text level over-sampling. However, the need for domain-specific background knowledge increases the difficulty of their application on other text classification tasks.

2.2. Complex network

Many systems in our world can be represented as networks whose nodes and edges are the system elements and their relations separately. With the complex relations among system elements, most networks of systems in the real world, like social relationships and author citation links, are complex networks. The complex network is a kind of graph that is neither regular nor random. Its global properties can not be inferred from the local interactions between certain node groups but emerge from the interactions of the whole elements (Cong, & Liu, 2014).

It is a hot field to study the natural language with complex networks, and a considerable number of researches have emerged in the last few years (Cong, & Liu, 2014; Arruda et al., 2016; Akimushkin et al., 2017; Garg, & Kumar, 2018). A network model of text is constructed with the language units (words or phrases) as the nodes connected by their interrelations (Cong, & Liu, 2014). The complex networks can develop their advantages when used to capture the high-level structural and semantic information of the text. It is also proved that complex networks can enhance the performance of various tasks in text analysis (Antiqueira et al., 2009; Amancio et al., 2012; Amancio, 2015; Arruda et al., 2016; Yan et al., 2019; Yan et al., 2020). Through complex networks, the words lose their order in the original sequences but retain their structural and semantic information. For example, the words that co-occur in the source text will also act as neighbors in the complex networks, and the highly used collocations of the text will behave as motifs (Goh et al., 2018).

According to the kind of relations between language units, the complex network of text can be divided into static and dynamic networks (Yan et al., 2020). For static networks, the language units are connected when they keep a semantic relationship, e.g., hypernymy, meronymy, and synonymy relationship. The edges of the dynamic networks are formed when two language units co-occur in the actual language use (Cong, & Liu, 2014). For example, in the co-occurrence complex network, a sub-structure of dynamic networks, two nodes are connected if they co-occur in the source text within a certain distance. In this paper, the proposed methods are all based on the co-occurrence complex network.

2.3. Random walk and graph embedding

Random walk is widely used in the analysis of complex networks. The idea of the random walk is simple and easy to be implemented. Let V represent all the nodes in a network model, and $v_i \in V$ represent a particular node. A random walk path with length k is generated after k times random jump with every jump aiming to an arbitrary neighbor of the node from the previous jump. In recent years, the random walk has been successfully used in node embedding of complex networks. DeepWalk (Perozzi et al., 2014) uses the random walk to generate several paths to learn the higher-order proximity between nodes. The target is to maximize the likelihood that a given node is centered around the $2m$ nodes along the random paths, where m nodes are to the left and m nodes are to the right. The node2vec (Grover, & Leskovec, 2016) preserves the ideas in DeepWalk. The main differences between DeepWalk and node2vec are that node2vec is based on a bias random walk and provides the options and trade-off between breadth-first searching (BFS) and depth-first searching (DFS). Walklets (Perozzi et al., 2016) also learn representations of vertices in a network through random walks. This model can capture multiple scales of relationships between networks, which is achieved by skipping some nodes during random walking. Many other models also capture the high-level information of networks through random walking (Li et al., 2016; Pan et al., 2016; Yang et al., 2016; Bojchevski et al., 2018; Xia et al., 2020), and the successful application of them verify that random walk is a powerful tool for analyzing complex networks.

Moreover, random walk is also used for over-sampling in imbalanced data classification. *Zhang and Li* (Zhang, & Li, 2014) propose a random walk over-sampling (RWO-Sampling) approach to balancing different class samples. This approach obeys the original minority class distribution and combines the random walk in the learned distribution with the original value for each attribute. *Roshanfekr et al.* (Roshanfekr et al., 2020) improve the RWO-Sampling method and develop a UGRWO-Sampling approach based on graphs to avoid the likelihood of over-fitting in RWO-Sampling. Our method is different from the above approaches in the following aspects. First, RWO-Sampling and UGRWO-Sampling are proposed for imbalanced numerical data with the assumption that the distribution of each attribute holds a Gaussian distribution. However, some vector representations for text like BoW are of high sparsity in some attributes and contain many zeros instead of keeping in line with the Gaussian distribution. Our method implements the random walk directly on the text complex network to avoid the distribution assumption, which is more suitable for text. Second, the random walk of the above approaches is directly based on the sample nodes, while our method is based on the features of each sample. Finally, our method is implemented before feature selection, while RWO-Sampling and UGRWO-Sampling are implemented after feature selection.

3. Model description

This section introduces the Polar Layer to transfer a document to multi-sampled random walk paths with a specific length. The output of the Polar Layer will be directly connected to a 2D CNN model, which is called NCNN in this paper.

3.1. Problem Definition

Let $D = \{d_1, d_2, ..., d_{|D|}\}$, where D is a collection of documents and d_i denotes the ith document in D. Given $L = \{l_1, l_2, ..., l_c\}, c << |D|$, we can assign every document in D to L to make the documents classified into c categories. We define $g = (V, E)$ as a network, where V is the set of nodes and $E \subseteq (V \times V)$ is the set of edges. Each edge $e \in E$ is the relationship between two nodes. We can use an ordered pair to represent this relationship, i.e., $e_{ij} = (v_i, v_j)$, and the strength of the relationship is denoted as ω_{ij}. In general, if there always exists the equivalence that $\omega_{ij} = \omega_{ji}$ for all node pairs that contain edges, we call G is undirected. Otherwise, if there exist node pairs with $\omega_{ij} \neq \omega_{ji}$, we call G is directed. Let $G = \{g_1, g_2, ..., g_n\}$ be a set of networks, where g_i is the representation of d_i. For convenience, we denote the relation between g_i and d_i as $g_i = \psi(d_i), g_i \in G, d_i \in D$.

There are different ideas on how to represent a document as a network, e.g., the dynamic network and the static network (Cong, & Liu, 2014; Yan et al., 2020). In this paper, unless specified, the following descriptions are all based on the co-occurrence dynamic network, where the edge, linking two nodes when they co-occur in the source text, is undirected. In the co-occurrence dynamic network, each node corresponds to a word, and the edges are the instance of the co-occurrence between nodes. For example, we show a network constructed from a short text "*This is just a toy example. Here we show how a complex network of a text is constructed.*" in Fig. 1. Note that the text is preprocessed: all letters are converted to lower case, and punctuations and special symbols are eliminated. In Fig. 1, the edge only refers to the co-occurrence of two nearest neighbors in the original text sequences, e.g., "this" and "is" are connected, "is" and "just" are connected, but "this" and "just" are not connected. Furthermore, because punctuations are removed in preprocessing, two words that are split by punctuation in the original text are still connected, e.g., "example" and "here" are connected with an edge. The nodes and edges are all unweighted, meaning that the weights of all nodes and edges are one regardless of the words' frequency in the source text.

There are two goals in this paper. First, we aim to define a representation of $g_i \in G$, from whom we can learn and extract features $F_i = \{f_{1i}, f_{2i}, \dots f_{n'i}\}$. The classifier then assigns the corresponding $d_i \in D$ to the correct category $l_j \in L$ according to F_i. Second, we aim to present a method to produce new samples from the text with label $l_j \in L$. And the extracted features $F_i^{'}$ from these new samples contain the representative information of the text to improve the performance of the classification task.

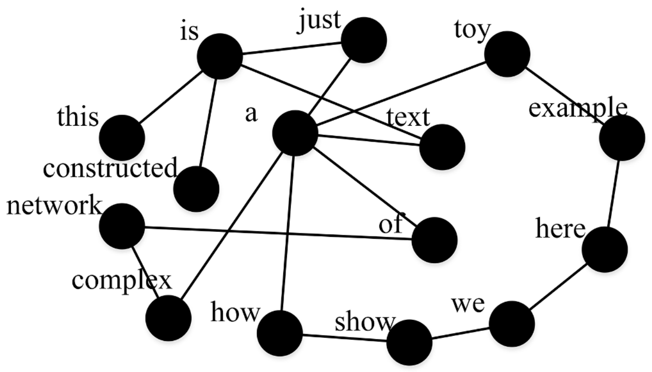


Fig. 1. An example: a complex network of text.

3.2. Start with random walk

There are several benefits of using random walks to extract the information of a document. A document needs space to store all the words (including punctuation marks sometimes), while it only needs to store the word pairs that co-occur in the source document when represented as a network. The sampling of a node in the random walk paths only acquires the information of neighbors or the second-order neighbors (the neighbors of neighbors), which makes it possible to sample on a large document with proper time costs. The high-level structural and semantic similarities can be extracted flexibly with different searching strategies (Perozzi et al., 2014; Grover, & Leskovec, 2016).

During a sampling process of random walk, given a start node u and the length l of random walk, let v_i denotes the node in site i along the path. After v_i is sampled, the next node v_{i+1} will be generated with probability

$$P(v_{i+1} = x | v_i = y) = \begin{cases} \dfrac{w_x}{\sum_{n \in N(y)} w_n}, & \text{if } x \in N(y) 0 \text{ , else} \end{cases}, \tag{1}$$

where $N(y)$ denotes the neighbors of node y, and w_x denotes the weight of x while searching. The most straightforward strategy is to assume that all neighbors have equal chances to be chosen for the next node, i.e., $w_x = 1, x \in N(y)$. Indeed, there are some options for us to get specific structural equivalence. We can use a local search around a node to capture the similarity of neighbors' states (Amancio et al., 2012). The direction of the local search can be controlled by P, which is determined by choice of w_x in Eqs. (1). Before going into the choice of w_x in this paper, we first introduce two kinds of similarities.

In Fig. 2, we show two kinds of similarities between documents. There are two documents doc_1 (left) and doc_2 (right). The local similarity between doc_1 and doc_2 may exist in particular nodes' neighbors, A_1 and A_2. If we only consider the nearest neighbors of nodes, the local similarity of A_1 and A_2 can be counted by the overlapping of nearest neighbors, e.g., B_1, B_2, and B_3. For example, some alternative words may have the same nearest neighbors. The short phrases like "source node," "source text," "source document" in this paper can also be written as "original node," "original text," "original document." In this case, "source" and "original" play the role of A_1 and A_2, and "node," "text," and "document" are equivalent to B_1, B_2, and B_3. In order to consider the high order of neighbors, the local search needs to extend far away from A_1 (A_2), i.e., the neighbors of neighbors. This kind of search strategy may reflect the similarity of certain topics. Take the topic of "pets" as an instance. This topic may describe a different aspect of pets, a dog and a cat all "like eating fishes," "play with a toy ball," and "sleep all day long," where we can think of a dog and a cat as A_1 and A_2, and think of the above three short phrases as the orange, green, and blue block, as shown in Fig. 2.

To consider two kinds of similarities mentioned above, we introduce the 2nd order random walk proposed in the literature (Grover, & Leskovec, 2016). Assume that a random walk leaves node s and stands on node m; it decides the next step to the neighbors of m and the probability distribution in Eqs. (1) is biased by α with the following equation

$$\alpha(m, e) = \begin{cases} \dfrac{1}{p} \text{ if } d_{es} = 0 \\ 1 \text{ if } d_{es} = 1 \\ \dfrac{1}{q} \text{ if } d_{es} = 2 \end{cases}, \tag{2}$$

where e is the next node; p and q are the hyper-parameters; $d_{es}$is the

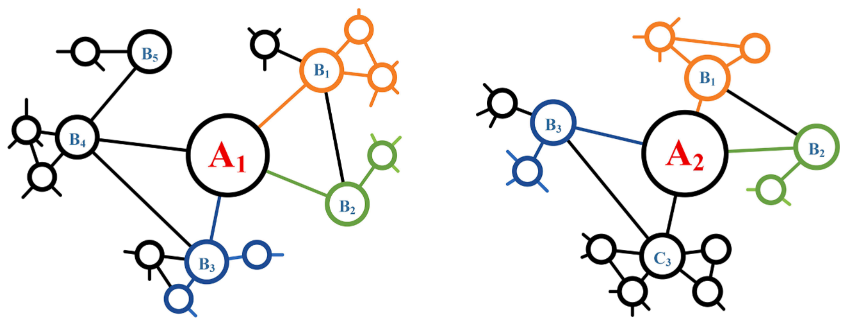


Fig. 2. Two kinds of similarities between documents: the similarity between neighbors (like B_1, B_2, and B_3) and the similarity between neighbors of neighbors (like the orange, green, and blue block). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

shortest step from node e to node s (e.g., $d_{es} = 0$ denotes that $e = s$). Now we go back to the choice of w_x mentioned above. By making the weight of x equal to α, i.e., $w_x = \alpha(y, x)$, the local search can be controlled to decide which similarity is considered in the probability distribution of Eqs. (1). We can adjust the search strategy by changing the value of p and q. The high value of p is to search high order of neighbors, and the high value of q is to search the nearest neighbors. As is shown in Fig. 3, the next step from m includes e_1, e_2, and e_3. According to the shortest path from e_1, e_2, and e_3 to s, it can be inferred that $d_{e1s} = 0$, $d_{e2s} = 2$, and $d_{e3s} = 1$, and thus $\alpha(m, e_1) = 1/p$, $\alpha(m, e_2) = 1/q$, and $\alpha(m, e_3) = 1$. Here, if we set $p > q = 1$, then node e_2 has a larger probability of being chosen as the next step, as is shown in Eqs. (1), and the random walk tends to step to high order of neighbors of s. Otherwise, if we set $q > p = 1$, the random walk will step around the nearest neighbors of s. The influence of p and q on classification performance will be studied in the experimental section.

3.3. Random path generation algorithm

Algorithm 1 shows the step of generating the random paths to rebuild the information of a text. p and q are set to get a trade-off between the nearest neighbors and high order of neighbors. The words of document d are labeled to reduce the memory cost. Every document is sampled with several random paths (with path number h), and h random paths are concatenated to be a matrix $\boldsymbol{M}$. $\boldsymbol{M}$ may be a $(1 \times hw)$ sequence or an $(h \times w)$ matrix. However, a sequence form of $\boldsymbol{M}$ can not consider different blocks of high-order of neighbors, so we choose an $(h \times w)$ matrix of $\boldsymbol{M}$ to be the output.

Algorithm 2 is to over-sample and expand the number of document samples. Note that $g = \psi(d_1, d_2)$ denotes to merge two networks of d_1 and d_2, including all the nodes and their edges. We can simply view it as two documents d_1 and d_2 being concatenated to form a more extended

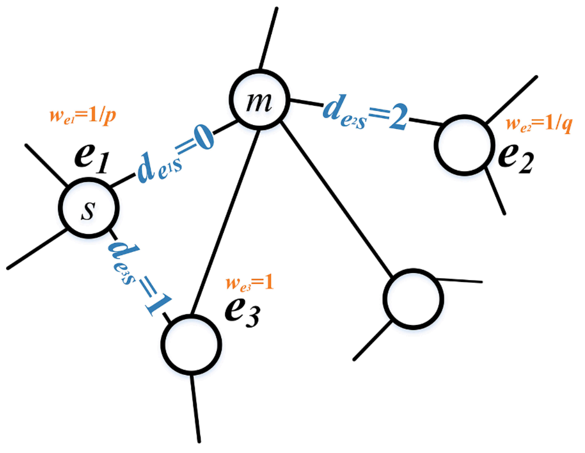


Fig. 3. Illustration of p, q, and d_{es}.

document d, and then represented as network g. Generally, d_1 and d_2 are asserted to belong to the same class, i.e., $d_1, d_2 \in l_i, l_i \in L$. We can use **Algorithm 2** several times to expand the samples to a decent number. A toy example of the above algorithms is described in Fig. 4, where path length $w = 5$, path number $h = 2$, and the document are sampled two times. In the following description, we denote the process to get the random walk paths through **Algorithm 1** or **Algorithm 2** as $\boldsymbol{M} = PathGen(g)$, and we can try this process several times and get different random paths, which is denoted as $\{\boldsymbol{M}_1, \boldsymbol{M}_2, ..., \boldsymbol{M}_n\} = PathGen^n(g)$.

3.4. Cross path convolution and Polar Layer

A random path generation will transform every document sample into a 2D matrix $\boldsymbol{M}$. One difficulty arises when deciding how to train this 2D matrix and extract features because the classical language model only supports a sequence of the document rather than a 2D matrix. To this end, we introduce a 2D convolution neural network (CNN) model to fit a 2D matrix. The 2D CNN model is widely used in computer vision tasks, in which a picture is stored as a 2D matrix. The 2D CNN model learns the features of 2D matrix by convolving the matrix with convolutional cores, which are smaller 2D matrices in width and height. Through convolving operation, the local features of the 2D matrix are assembled into many matrices with smaller width and height than the original 2D matrix. When the convolution is applied to text, the text CNN model will convolve a matrix containing only width from a sequence of words (Kim, 2014). Furthermore, the words of text are represented as vectors, and the input of CNN model for text is a sequence of word vectors. In this paper, the proposed network-based CNN (NCNN) model is specifically designed to process a 2D matrix $\boldsymbol{M}$, in which each value is also represented as vectors and thus yield $\boldsymbol{M}' \in \mathbb{R}^{h \times w \times d}$, where d is the dimensionality of vectors. In Fig. 5, we show the difference between the input of text CNN (a) and NCNN (b), where the x-axis denotes the direction of text sequence and random paths, z-axis denotes the dimensionality of word vectors. The input of NCNN has y-axis, which denotes the number of random walk paths.

Based on the input of NCNN, we propose the cross paths convolution (CPC) to learn the high-order similarity of neighbors. The CPC operation is that convolution is convolved among more than one random path, as shown in Fig. 6. The high-order similarity of neighbors may be extracted from more than one random path, and CPC can learn this information by assembling different random walk paths with Conv Core. In Fig. 6, different random paths starting from node 1 are generated and transformed into a 2D matrix. Then a Conv Core assembles the similarity information of orange, blue, and green blocks.

The CPC operation can be easily achieved using 2D CNN model, except for the first Convolution Layer because the input of NCNN has an extra dimension of the vectors' dimensionality. Here, we propose a Polar Layer as the medium between the input and the first Convolution Layer of the 2D CNN model. The Polar Layer carries out a convolution operation between a filter $\boldsymbol{m} \in \mathbb{R}^{h \times w_m \times d}$ and a matrix $\boldsymbol{M}' \in \mathbb{R}^{h \times w \times d}$, which

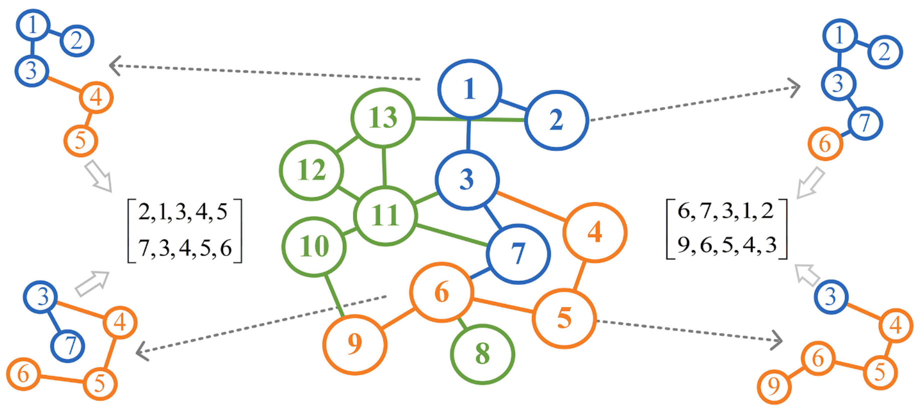


Fig. 4. A toy example of rebuilding information of a text and over-sampling. The path length $w = 5$, path number $h = 2$, and the document are sampled two times.

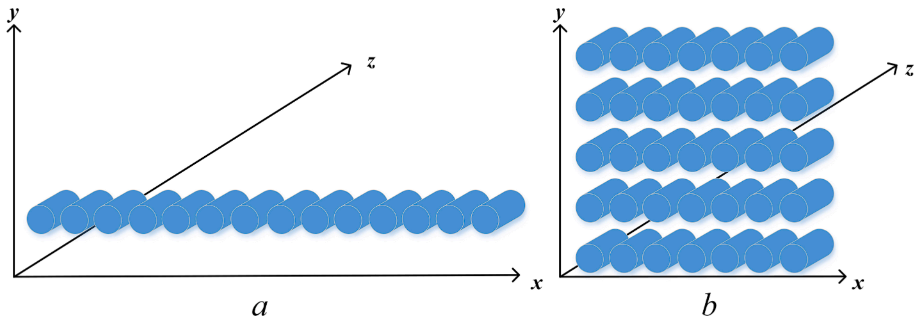


Fig. 5. The input of text CNN (*a*) (Kim, 2014) and NCNN (*b*).

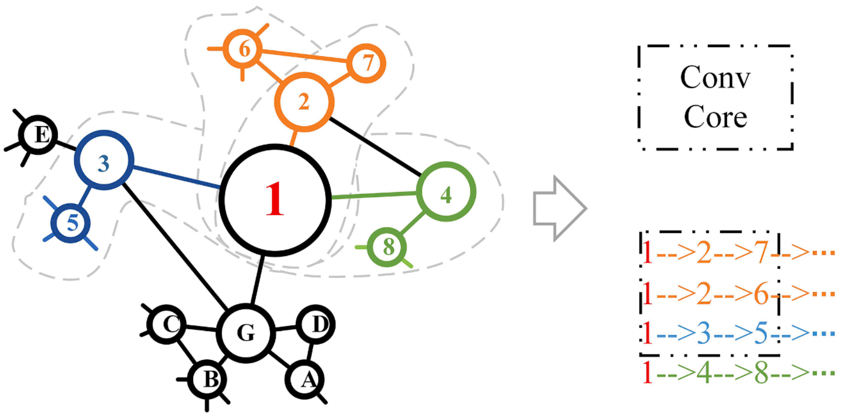


Fig. 6. Cross paths convolution. The convolution kernel (Conv Core) convolves across different paths along the direction of random paths, e.g., the Conv Core convolves three different paths, starting from node 1.

produces another matrix $\boldsymbol{O} \in \mathbb{R}^{h\times w_o}$:

$$\boldsymbol{O}(:,i) = \sum_{j=1}^{d} \boldsymbol{m}(:,j)^{\circ}\boldsymbol{M}'(:,i:i+w_m-1,j). \tag{3}$$

In Eqs. (3), "$\circ$" denotes the *Hadamard* product, and the values in "()" denote the indexes of the corresponding dimension. Note that ":" denotes indexing all values. w_o is determined by the convolution operation. The structure of the Polar Layer is shown in Fig. 7, where the x-axis denotes the direction of this operation, the y-axis denotes the height (h) of $\boldsymbol{M}$', and the z-axis denotes the dimension (d) of the vectors of the nodes. From Fig. 7, we can observe that the z-axis of the output from the Polar Layer is flattened, yielding a proper format for the processing of the Convolution Layer in the 2D CNN model. In the following description, we represent the operation of the Polar Layer as $\boldsymbol{O} = \boldsymbol{m}\otimes\boldsymbol{M}'$.

3.5. The structure of NCNN

Fig. 8 shows the structure of the Network-based Convolution Neural Network (NCNN) model. Note that the shape marked in parentheses is the output shape of each layer, where N denotes the number of samples (with length S_l) and h_i, w_i are determined by the convolution or max-pooling operation. We can view a Polar Layer output as the input of a 2D Convolution Layer (Conv2D) with only one channel (generally, the matrix of a picture has three channels). After two Conv2D layers, two fully-connected layers are connected to produce the classification results.

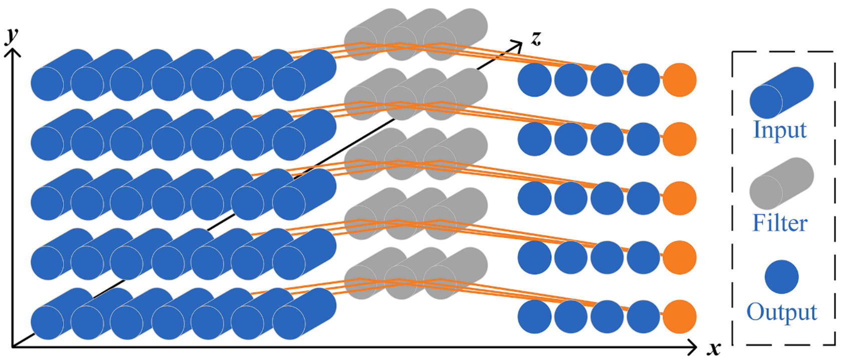


Fig. 7. Structure of Polar Layer. For $\boldsymbol{M}$, $h = 5$, $w = 7$. For the Conv Core (denoted as Filter in this figure), $h = 5$, $w_m = 3$. As a result, $w_o = 5$.

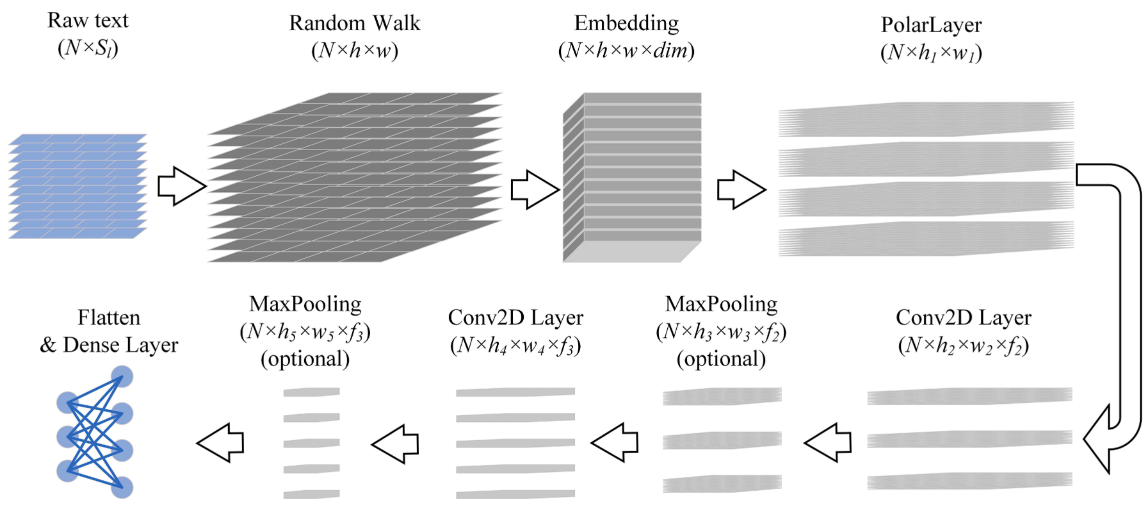

Raw text ($N\times S_l$)
Random Walk ($N\times h\times w$)
Embedding ($N\times h\times w\times dim$)
PolarLayer ($N\times h_1\times w_1$)
Flatten & Dense Layer
MaxPooling ($N\times h_5\times w_5\times f_3$) (optional)
Conv2D Layer ($N\times h_4\times w_4\times f_3$)
MaxPooling ($N\times h_3\times w_3\times f_2$) (optional)
Conv2D Layer ($N\times h_2\times w_2\times f_2$)

Fig. 8. Structure of NCNN. N denotes the number of documents (with length S_l), h_i, w_i are determined by the convolution or max-pooling operation, and *dim* denotes the dimension of node vectors. Note that *fi* is the number of convolution kernels in the last layer.

3.6. An electing strategy for predicting

The random walk sampling can participate in not only feature extracting but also predicting. In predicting tasks like text classification, given a document d, a prediction model is used to predict the label l of d. Generally, the prediction model calculates the probability $p(l_i)$ of d belonging to label $l_i \in L$, and then the label with maximum probability is selected as the predicted label, i.e., $l = argmax\{(p(l_i))\}, l_i \in L$. In this paper, the last layer of the CNN model for classification produces the probability distribution of all labels. The label with the maximum probability is the final selected label as the prediction result.

ALGORITHM 1 Path Generation for Source Text

```
Input $\boldsymbol{p}$, $\boldsymbol{q}$, document $\boldsymbol{d}$, path length $\boldsymbol{w}$, path number $\boldsymbol{h}$
Output $\boldsymbol{M}_{(h\times w)}$
1: Represent $d$ as $g$, i.e., $g = \psi(d)$
2: Initialize path_num, length as 0; initialize $M$ as null; initialize path as null
3: while path_num < $h$ do
4:   if length is 0 then
5:     randomly choose a start node $s$
6:     append $s$ to path
7:   else
8:     choose the next node $v$ using (1) and (2) with $p$, $q$, $g$
9:     length = length + 1
10:    append $v$ to path
11:    if length is $w$ then
12:      path_num = path_num + 1
13:      append path to $M$
14:      initialize path as null; initialize length as 0
15:    end if
16:  end if
```

(continued on next column)

(continued)

ALGORITHM 1 Path Generation for Source Text

```
17:    output: $\boldsymbol{M}_{(h\times w)}$
18:end while
```

ALGORITHM 2 Path Generation for Over-sampling

```
Input $\boldsymbol{p}$, $\boldsymbol{q}$, document $\boldsymbol{d_1}$, and $\boldsymbol{d_2}$, path length $\boldsymbol{w}$, path number $\boldsymbol{h}$
Output $\boldsymbol{M}_{(h\times w)}$
1. Represent $d_1$ and $d_2$ as $g$, i.e., $g = \psi(d_1, d_2)$
1. Follow steps 2 and 3 of Algorithm 1 to get the output matrix $M$
```

Next, we describe an electing strategy for predicting classification results. By using the random walk sampling, we generate n samples of d. Then, the corresponding label is predicted for every sample, and the final predicted label for d is chosen from them. Fig. 9 shows the process of the electing strategy. We introduce two strategies to obtain the final label: ES-mean and ES-max.

ES-mean strategy first gets the probability that n samples are predicted to belong to label l_i, and then uses Eqs. (4) to calculate the probability of $\boldsymbol{M}_k$ belonging to label l_i, where $\boldsymbol{M}_k \in \{\boldsymbol{M}_1, \boldsymbol{M}_2, ..., \boldsymbol{M}_n\} = PathGen^n(g), g = \psi(d)$.

$$p(l_i) = \frac{1}{n}\sum_{k=1}^{n} p_k(l_i) \tag{4}$$

ES-max strategy first gets predicted labels from n samples and then chooses the most occurred label as the final label of document d, as Eqs. (5) shows, where $x_{ki} = 1$ if $\text{argmax}\backslash\{(p_k(l_i))\} = l_k$, and $x_{ki} = 0$

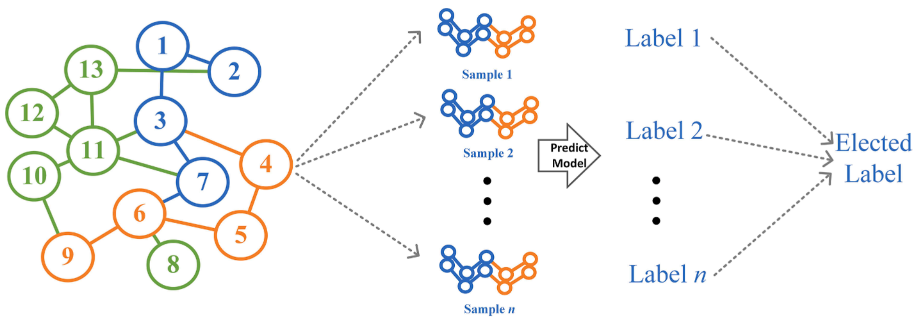


Fig. 9. The process of electing strategy. The *Es-mean* strategy chooses the elected label based on the mean of the probability of each label, and the *ES*-max strategy chooses the most predicted label as the elected label.

otherwise.

$$p(l_i) = \frac{\sum_k x_{ki}}{\sum_k \sum_j x_{kj}} = \frac{\sum_k x_{ki}}{n} \tag{5}$$

We can see the electing strategy as the assembling of the output from the last layer of CNN. By using electing strategy, the prediction result is decided after many trials, and the final result is the most conservative choice from multiple attempts.

4. Data and experimental results

In this section, we introduce two datasets and test the performance of the proposed model by comparing it with different baselines in text classification. The first dataset is *Reuters 21578*, which contains a highly uneven distribution of samples in different classes. The other dataset, *WebKb*, is the balanced dataset. With the same idea proposed in literature (Wu et al., 2014), we manually transform the dataset into imbalanced dataset by eliminating some samples in certain classes.

4.1. Data description

Reuters 21578 is manually collected and classified from Reuters Ltd. The source dataset suffers from a very skewed class distribution. So we choose a refined dataset of *Reuters 21578*, which is called *R52* (Craven et al., 2003). In Table 1, we show part of the classes in *R52*, in which the number of samples in “earn” is ten times more than in “bop.” Moreover, some classes like “nickel,” “platinum” have samples less than 5. So *R52* still has an uneven class distribution, which may cause inefficiency training in some classes.

WebKb is a series of texts from web pages collected by the World Wide Knowledge Base project (Craven et al., 2003). This dataset contains four classes, each containing enough samples for training because applying many approaches for solving data imbalance directly on the source datasets can only get a very limited improvement on performance. So, we eliminate 90% of the samples in the “project” class, which contains the minimum number of samples in the source data, to ensure the dataset is imbalanced. In Table 2, we show the number of samples in each class before and after eliminating samples.

4.2. Experimental setup

All the methods used in this paper are implemented in *Windows 10* based on *Python 3.7* with *TensorFlow* 2.0, which is used to build the neural network. All the models and methods are run on a desktop computer that is configured for *Intel® Core™ i5-6600 CPU @ 3.30 GHz, RAM 16.0 GB, GTX2060 6 GB*.

We choose Accuracy, F1-measure, Precision, and Recall as the evaluation measures. Table 3 shows the details of these metrics. In this paper, F1-measure, Precision and Recall are calculated considering their macro average and micro average because the datasets contain more than two labels (multi-class). The micro average calculates metrics globally by counting the total TP, FN, and FP, while the macro average calculates metrics for each label and gets their unweighted mean. Note that the value of Accuracy, micro F1-measure, micro Precision, and micro Recall are equal in multi-classification. So the results will only show Accuracy, macro F1-measure, macro Precision, and macro Recall.

We use the following models to be the classifiers. Note that some models are only for vector representation of documents, and these models all use SVM as classifiers. Table 4 describes the structures of all the classifiers.

Table 2
Number of samples before and after eliminating in *WebKb*.

Class	student	faculty	course	project	
				before eliminating	after elimination
# train docs	1097	750	620	336	33
# test docs	544	374	310	168	168
Total # docs	1641	1124	930	504	201

Table 3
Definition of Accuracy, F1-measure, Recall, and Precision.

	Predicted belong to l_i	Predicted not belong to l_i
Actual belong to l_i	TP_i	FN_i
Actual not belong to l_i	FP_i	TN_i
Accuracy_i	$\frac{TP_i + TN_i}{TP_i + FP_i + TN_i + FN_i}$	
F1-measure_i	$\frac{2 \times P_i \times R_i}{P_i + R_i}$	
Recall_i	$\frac{TP_i}{TP_i + FN_i}$	
Precision_i	$\frac{TP_i}{TP_i + FP_i}$	

Note: $R_i = Recall_i$, $P_i = Precision_i$

Table 1
Number of samples in *R52*.

Class	earn	acq	crude	trade	cocoa	bop	potato	nickel	platinum
# train docs	2840	1596	253	251	46	22	2	3	1
# test docs	1083	696	121	75	15	9	3	1	2
Total # docs	3923	2292	374	326	61	31	5	4	3

Table 4
Structure of classifiers.

Model	CNN	BoW.tf, BoW.tfidf
Feature representation	Embedding	Bag of words
Classifier	Convolution Layer (with: Kernel size: 5 Stride: 1 Filters: 50) MaxPooling Layer Fully Connected Layer (with: Filters: 200) Dropout Fully Connected Layer	SVM (with: c: {0.1, 0.5, 1, 2, 5, 10})

CNN is a convolution neural network for text classification. The architecture of CNN is similar to the model proposed by (Kim, 2014) with fine-tuning in parameters.

BoW.tf is a bag-of-words representation model with term frequency as the features.

BoW.tfidf (Naderalvojoud et al., 2015) is a bag-of-words representation model with *TFIDF* values as the features.

We choose the following methods to solve data imbalance and their combination with the above classifiers as the comparison methods. Moreover, their combination is represented as a "method-classifier" form, e.g., a CNN classifier combined with RO will be described as "RO-CNN."

RO: Random over-sampling (RO) method is a method for directly duplicating the samples in certain classes of the datasets.

SMOTE (Chawla et al., 2002): Synthetic minority over-sampling technique (SMOTE) generates new synthetic examples for minority classes, which is different from RO.

PNF (Naderalvojoud et al., 2015): PNF is a term weighting approach to solve data imbalance problems instead of over-sampling.

BDSK (Song et al., 2016): BSDK is a bi-directional sampling method based on K-means and SMOTE (for over-sampling), which over-samples minority classes and under-samples the majority classes to a certain number. In this paper, we only use its over-sampling strategy because the under-sampling leads to performance reduction.

RWO (Zhang, & Li, 2014): A random walk over-sampling method for numerical data. In this paper, we apply this approach to the BoW representation of text.

TextGen (Shaikh et al., 2021): A background-based method for over-sampling. This method generates new texts through the generation model after a fine-tuning based on the imbalanced datasets. In this paper, the generation model is based on the model from https://minimaxir.com/2018/05/text-neural-networks/.

The proposed model is represented as "NCNN," and its combination with an electing strategy is represented as "n-NCNN" for *ES-mean* and "x-NCNN" for *ES-max*.

We use the *sklearn* module to calculate the Accuracy, F1-measure, Recall, and Precision value and use its *svm.SVC* function to implement SVM with its default parameters except for *c*. We search *c* in {0.1, 0.5, 1, 2, 5, 10} to choose the best performance as the final results.

Note that the Embedding layer of NCNN adopts a pre-trained embedding from a large corpus, including *R52* and *WebKb,* to avoid changing the feature space of the original data when training the random walk paths. The data is preprocessed by removing the stop words for BoW.tf and BoW.tfidf. For CNN and NCNN, the data will keep all the words. The training epochs of CNN, RO-CNN, TextGen-CNN, and NCNN are set to 10.

4.3. Performance and comparison

For *R52*, the over-sampling methods are only applied to the classes of training data with the number of samples fewer than 400. These classes are over-sampled to make the number of samples reach 400. We keep the classes with the number of samples fewer than 400 as it is. Though this may not completely solve the problem of data imbalance, it can get good results without too many samples to be trained, which will need a much higher time cost. We also tried under-sampling the majority classes to 400, but results showed that this would cause a performance reduction. For *WebKb*, the over-sampling methods are applied to all the classes whose number of samples is fewer than 1097 (the number of samples in the majority class). Table 5 and Table 6 show the details of the increased number of the expanded data. Table 5 only lists four classes to show the difference between classes that are over-sampled and those not over-sampled.

Table 7 and Table 8 show the results achieved with different methods. The proposed methods, including NCNN, n-NCNN, and x-NCNN, are implemented with $p = 1$ and $q = 1$ (equal to a random walk). Because the RWO, SMOTE, PNF, and BDSK methods can only be applied to numerical data, CNN only has the combination of RO and TextGen, i.e., RO-CNN and TextGen-CNN. Moreover, PNF is only combined with BoW.tf because the PNF method is a term weighting method different from TFIDF. PNF takes the category membership into account, while TFIDF does not. We show the results of CNN in the first line of Table 7 and Table 8: CNN achieves the best performance among CNN, BoW.tf, and BoW.tfidf when no over-sampling strategy is implemented on the original data. Moreover, the results show that all the over-sampling strategies can get a performance improvement.

The deep learning methods, like RO-CNN, TextGen-CNN, NCNN, n-NCNN, and x-NCNN, all achieve excellent results in Accuracy. However, RO-CNN and TextGen-CNN perform poorly in macro F1-measure, macro Recall, and macro Precision for *R52*. For *WebKb*, TextGen-CNN gets a better promotion compared with RO-CNN. As mentioned above, TextGen-CNN is a knowledge-based method for over-sampling, which learns the styles of the samples belonging to a class and generates new texts to increase the number of samples based on a powerful text generation model. However, we can observe that this method can not work well on *R52*. *R52* contains many minority classes with a few samples (17 classes contain less than ten samples), which can not guarantee that a generation model is well-trained based on these samples. In Table 9, the mean values of F1-measure with respect to different sizes of classes for *R52* are listed. The value for "C1-10" of TextGen-CNN is 0.0000, denoting that the generation model generates wrong samples for the classes with too few samples ("C1-10" denotes the classes with the number of samples less than 10). We can view the Accuracy measure as the overall accuracy of all the classes, where the minority classes have little effect on the final results because the number of samples will influence the effect. On the contrary, the macro mean value of metrics, including macro F1-measure, macro Recall, and macro Precision, take every class into equal account, regardless of the number of samples. So the macro mean value can reflect how a method for the problem of imbalanced data performs on the minority classes. The poor behavior of RO-CNN in the macro mean value of metrics indicates that a RO strategy will not always work because the simple copy of samples may not correctly reflect the real feature space of training data.

For the methods based on the BoW model, SMOTE performs better than RO in Accuracy, macro F1-measure, and macro Recall. BDSK is also a SMOTE-based method, which uses K-means to over-sample the samples in similarity feature distribution. Because SMOTE can only be

Table 5
Changes in expanded training data of *R52*.

Class	# Original data	# Expanded data	# Changes
earn	2840	2840	+ 0
acq	1596	1596	+ 0
crude	253	400	+ 147
…	…	…	…
platinum	1	400	+ 399
total	6532	24,436	+ 17,904

Table 6
Changes in expanded training data of *WebKb*.

Class	# Original data	# Expanded data	# Changes
student	1097	1097	+ 0
falcuty	750	1097	+ 347
course	620	1097	+ 477
project	33	1097	+ 1064
total	2500	4388	+ 1888

Table 7
Results achieved with different metrics for *R52*.

Measure	Accuracy	macro F1	macro Recall	macro Precision
CNN	0.8824	0.3503	0.3440	0.4051
RO-CNN	0.9194	0.5666	0.5735	0.5923
TextGen-CNN	0.8934	0.3503	0.4120	0.3568
RO-BoW.tf	0.8878	0.5426	0.4893	0.6877
RO-BoW.tfidf	0.8882	0.5418	0.4882	0.6898
SMOTE-BoW.tf	0.9058	0.6097	0.5674	0.7215
SMOTE-BoW.tfidf	0.9034	0.6015	0.5591	0.7184
PNF-BoW.tf	0.9280	0.6439	0.6024	0.7375
BDSK-BoW.tf	0.9003	0.5688	0.5272	0.6856
BDSK-BoW.tfidf	0.8984	0.5761	0.5274	0.7185
RWO-BoW.tf	0.8941	0.5628	0.5133	0.6897
RWO-BoW.tfidf	0.6928	0.5776	0.5381	0.6969
NCNN	0.9264	0.7059	0.7286	0.7141
n-NCNN	**0.9389**	0.7489	0.7480	**0.7776**
x-NCNN	0.9380	**0.7545**	**0.7719**	0.7765

Table 8
Results achieved with different metrics for *WebKb*.

Measure	Accuracy	macro F1	macro Recall	macro Precision
CNN	0.8209	0.6708	0.7082	0.7526
RO-CNN	0.8266	0.6929	0.7179	0.8260
TextGen-CNN	0.8524	0.7915	0.7820	0.8562
RO-BoW.tf	0.8195	0.6610	0.7001	0.8746
RO-BoW.tfidf	0.8188	0.6607	0.7001	0.8746
SMOTE-BoW.tf	0.8259	0.6995	0.7184	0.8368
SMOTE-BoW.tfidf	0.8231	0.6926	0.7146	0.8425
PNF-BoW.tf	0.8653	0.6918	0.7357	0.6573
BDSK-BoW.tf	0.8202	0.6670	0.7031	0.8754
BDSK-BoW.tfidf	0.8209	0.6726	0.7052	0.8344
RWO-BoW.tf	0.8181	0.6575	0.6986	0.6243
RWO-BoW.tfidf	0.8173	0.6570	0.6979	0.6238
NCNN	0.8510	0.8189	0.8103	0.8405
n-NCNN	**0.8829**	**0.8595**	**0.8505**	**0.8779**
x-NCNN	0.8720	0.8471	0.8387	0.8666

Table 9
Mean values of F1-measure with respect to different sizes of classes for *R52*.

Measure	C1-10	C10-100	C100-500	C500-
CNN	0.0588	0.3894	0.7929	0.9710
RO-CNN	0.1644	0.7327	0.8237	0.9699
TextGen-CNN	0.0000	0.4667	0.7662	0.9745
RO-BoW.tf	0.1760	0.6947	0.7651	0.9382
RO-BoW.tfidf	0.1760	0.6934	0.7641	0.9394
SMOTE-BoW.tf	0.2608	0.7605	0.8058	0.9511
SMOTE-BoW.tfidf	0.2609	0.7461	0.7990	0.9512
PNF-BoW.tf	0.3197	0.7720	0.8814	0.9580
BDSK-BoW.tf	0.2079	0.7358	0.7764	0.9480
BDSK-BoW.tfidf	0.2021	0.7228	0.7883	0.9495
RWO-BoW.tf	0.2021	0.7136	0.7798	0.9424
RWO-BoW.tfidf	0.2608	0.7274	0.8231	0.5184
NCNN	0.4796	0.8038	0.8171	0.9744
n-NCNN	**0.5500**	0.8318	**0.8644**	0.9761
x-NCNN	0.5206	**0.8645**	0.8485	**0.9771**

Note: “C1-10” denotes all the classes with the number of samples from 1 to 10 (not included); “C500-” denotes all the classes with the number of samples>500 (included).

applied to numerical data, text data must be transferred into numerical form, the performance of which will be affected by feature extraction. Also, the transformation from text data into numerical data may lose valuable information. NCNN is to over-sample directly on the text data, which catches the high-level semantical information of the text. The performance of NCNN gets a significant improvement compared with RO and SMOTE. RWO, an improved over-sampling way with the same synthetic sampling idea as SMOTE, is a random walk based method. In section 2.3, we compare its random walk with our proposed method: the random walk of RWO is different from NCNN in many aspects. The results show that SMOTE performs better than RWO. Furthermore, TFIDF may be more suitable for RWO than TF features.

The term weighting methods, like TFIDF and PNF, consider the prior term weighting information in the training data and overcome the impact of data imbalance. TFIDF is a standard weighting strategy that has been proved to be effective in many classification tasks. However, TFIDF does not take into account the category information. So it still needs the over-sampling strategy to get a considerable performance. PNF considers the difference between classes, and from the results, we can observe that PNF achieves better performance than TFIDF. Also, PNF gives a significant promotion on Accuracy for both two datasets. But the promotion of minority classes is limited, e.g., F1-measure of “C1-10” (*R52*) and “project” (*WebKb*) in Table 8 and Table 9.

The proposed methods, including NCNN, x-NCNN, and n-NCNN, all get good results. The best value of Accuracy, macro F1-measure, macro Recall, and macro Precision are all obtained by the proposed methods. The proposed methods are marked in *italics* in Tables, and the best values are marked in **bold**. Though RO-CNN also over-samples on the text data instead of numerical data, the F1-measure between NCNN and CNN is profoundly different. We may attribute this difference to the high-level operation of NCNN. NCNN over-samples more extending samples through **Algorithm 2**, which produces profoundly different samples even from the same couple of samples. The variety of samples can help with reducing the impact of overfitting. An electing strategy further improves the performance of NCNN, certifying its effectiveness in classification tasks.

We also explore the F1-measure of every class and the values of each method. We aim to find out what affects the F1-measure in Table 7 and Table 8. We give the results in Fig. 10 (for *R52*) and Table 10 (for *WebKb*). For *R52*, because it is lengthiness to present the results of 52 classes, we present the number of classes whose F1-measure values are in a certain range. The number of classes with F1-measure value in range “>0.9”, “>0.7&<=0.9”, “>0.4&<=0.7”, “>0&<=0.4” and “=0” is counted separately. And then, the values are normalized into 0–1, as is shown in Fig. 10. A high proportion in the bottom region of the column indicates a high performance of the corresponding method. The results clearly show the advantage in range 0.7–0.9 of NCNN, n-NCNN, and x-NCNN. In Table 9, we show the mean values of F1-measure with respect to different sizes of classes for *R52*. Results show that the proposed method is competitive in majority classes with other methods and has an advantage in minority classes. In Table 10, F1-measure value with respect to each class for *WebKb* is presented. These results once again confirm the advantage of the proposed method in minority classes.

4.4. *The influence of p and q*

Next, we analyze the influence of p and q, which are two hyper-parameters that change the choice of the next step during the generation of a random path, as described in Eqs. (2).

Fig. 11 and Fig. 12 show the results from NCNN, n-NCNN, and x-NCNN for *R52* and *WebKb* separately, with the same setup as the above experiments. The experiment is implemented several times with different p and q: we first fix p to 1 and change q from 1 to 5, then fix q to 1 and change p from 1 to 5.

From the results, we can observe that value of metrics achieved from NCNN, n-NCNN, and x-NCNN for *R52* follow an M−shaped trend with

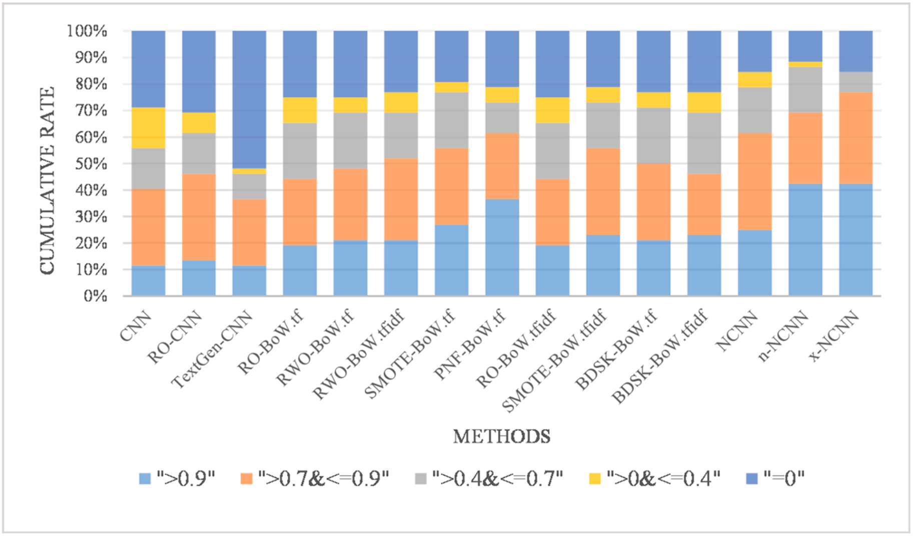


Fig. 10. F1-measure of each method and its distribution. A high proportion in the bottom region of the column indicates a high performance of the corresponding method.

Table 10
F1-measure value with respect to each class for *WebKb*.

Measure	student	faculty	course	project
CNN	0.8953	0.8227	0.8983	0.0670
RO-CNN	0.8933	0.8152	0.9219	0.1413
TextGen-CNN	0.9056	0.8301	0.9201	0.5105
RO-BoW.tf	0.8819	0.8000	0.9501	0.0118
RO-BoW.tfidf	0.8813	0.7981	0.9516	0.0118
SMOTE-BoW.tf	0.8842	0.8024	0.9501	0.1613
SMOTE-BoW.tfidf	0.8844	0.7986	0.9453	0.1421
PNF-BoW.tf	**0.9611**	0.8270	**0.9790**	0.0000
BDSK-BoW.tf	0.8813	0.8000	0.9516	0.0351
BDSK-BoW.tfidf	0.8829	0.8000	0.9499	0.0575
RWO-BoW.tf	0.8813	0.7972	0.9516	0.0000
RWO-BoW.tfidf	0.8806	0.7958	0.9516	0.0000
NCNN	0.8893	0.8056	0.9305	0.6500
n-NCNN	0.9057	**0.8549**	0.9506	**0.7266**
x-NCNN	0.8977	0.8367	0.9477	0.7063

the change of p / q. The shallow point is located in $p / q = 1 / 1$ of the x-axis, indicating that the purely random walk can not achieve the best in capturing the high-level structural and semantic information of the text. Moreover, the peaks exist in the points with p larger or smaller than q. For *WebKb*, we can also observe that the best performance is achieved with p larger or smaller than q, though the trend is different from *R52*.

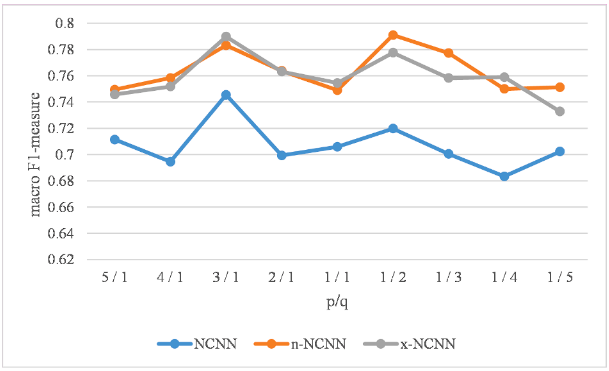
We show the above results to argue that a targeted walk aimed at the similarity of nearest neighbors or faraway neighbors can better capture the high-level information instead of the purely random walk. For *R52*, the best performance occurs in x-NCNN when $p / q = 3 / 1$. The value of Accuracy is 0.9474, and the value of macro F1-measure is 0.7899. For *WebKb*, the best performance occurs in n-NCNN when $p / q = 1 / 5$. The value of Accuracy is 0.8966, and the value of macro F1-measure is 0.8780.

Though this paper use $p = 1$ and $q = 1$ to get all the results in the experimental section, the influence of p and q can shed light on how to fine-tune the two hyper-parameters. We can try many times with different p and q, and use validation data to judge which set of values is the best. We can also skip some choices, like $p = 1$, $q = 1$, and $p = 2$, $q = 1$, according to the influence trend mention above. Then, we choose p and q that get best validation results to predict for the test data.

5. Conclusion

This study proposed a network-based feature extraction model for text analysis to deal with imbalanced text data. The random walk was introduced to generate new synthetic samples. An NCNN model was then proposed to combine the novel feature extraction model with the classical deep learning method (CNN). A Polar Layer is designed to fit the data from random walk paths to CNN.

By comparing with RO, SMOTE, BDSK, PNF, RWO, TextGen combined with CNN and other classification models, we proved the effectiveness of the proposed NCNN model on the imbalanced data. We also introduced an electing strategy for predicting to improve accuracy,



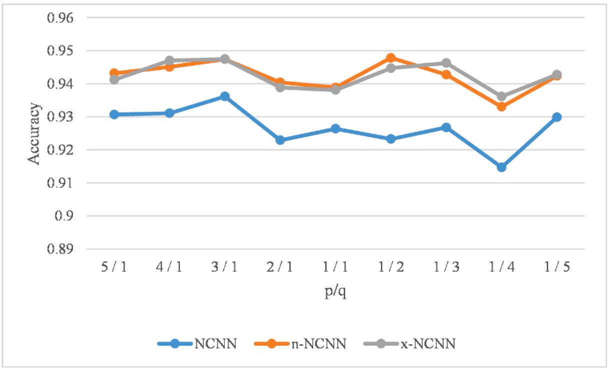
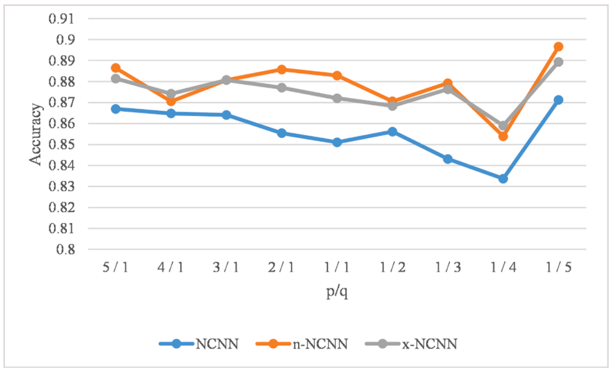


Fig. 11. The Accuracy and macro F1-measure for *R52* achieved with different p and q.



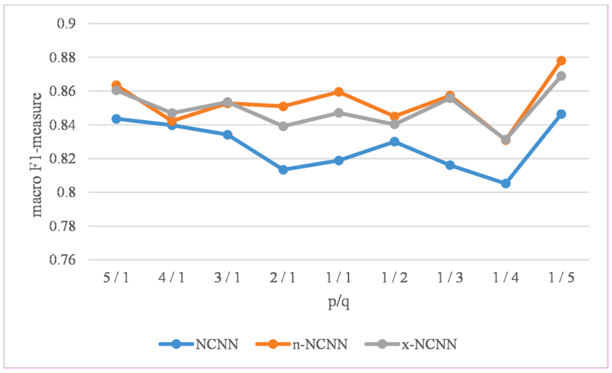


Fig. 12. The Accuracy and macro F1-measure for *WebKb* achieved with different *p* and *q*.

which is useful in some tasks like classification, especially when the number of data is limited. The results showed that the NCNN model with an electing strategy achieved a significant improvement in performance.

Exploration was also conducted to find the best combination of p and q for the proposed model (NCNN, n-NCNN, and x-NCNN). We observed that the trend following the change of p / q presents an M-shaped curve. The best performance occurred when $p / q = 3 / 1$ for *R52* and $p / q = 1 / 5$ for *WebKb*, indicating that a targeted walk could capture the high-level information better than what a random walk could do.

The limitation of our study is that the random walk paths lose the sequence information of the original text, which originates in the way of network construction. The future work will concentrate on combining the original text with the random walk paths to introduce the sequence order, and then apply the model to more broad text analysis tasks.

CRediT authorship contribution statement

Keping Li: Writing – original draft, Supervision, Project administration, Funding acquisition. **Dongyang Yan:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Software, Formal analysis, Investigation, Resources. **Yanyan Liu:** Validation, Resources, Writing – review & editing. **Qiaozhen Zhu:** Validation, Data curation, Visualization, 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.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant Nos.71942006, 71621001), and the Research Foundation of State Key Laboratory of Railway Traffic Control and Safety, China, Beijing Jiaotong University (Grant No. RCS2021ZT001). The authors sincerely thank Prof. Tiezhu Jin for his meticulous polishing of the article.

References

Akimushkin, C., Amancio, D. R., & Oliveira, O. N., Jr. (2017). Text authorship identified using the dynamics of word co-occurrence networks. *PLoS ONE, 12*(1), Article e0170527.

Amancio, D. R., Oliveira, O. N., Jr., & Costa, L. da. F. (2012). Structure–semantics interplay in complex networks and its effects on the predictability of similarity in texts. *Physica A: Statistical Mechanics and its Applications, 391*, 4406–4419.

Amancio, D. R., Oliveira, O. N., Jr., & Costa, L. D. F. (2012). Structure–semantics interplay in complex networks and its effects on the predictability of similarity in texts. *Physica A: Statistical Mechanics and its Applications, 391*(18), 4406–4419.

Amancio, D. R. (2015). Probing the topological properties of complex networks modeling short written texts. *PLoS One, 10*(2), Article e0118394.

Antiqueira, L., Oliveira, O. N., Jr., Costa, L. D. F., & Nunes, M. D. G. V. (2009). A complex network approach to text summarization. *Information Sciences, 179*(5), 584–599.

Arruda, H. F. D., Costa, L. D. F., & Amancio, D. R. (2016a). Topic segmentation via community detection in complex networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science, 26*(6), Article 063120.

Arruda, H. F. D., Costa, L. D. F., & Amancio, D. R. (2016b). Using complex networks for text classification: Discriminating informative and imaginative documents. *Europhysics Letters, 113*(2), 28007.

Bojchevski A., Shchur O., Zügner D., & Günnemann S. (2018). NetGAN: Generating graphs via random walks, *35th International Conference on Machine Learning (ICML)*, 2018. vol. 2, pp. 973-988.

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Collobert, R., Weston, J., Bottou, L., & Karlen, M. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research, 12*(7), 2493–2537.

Cong, J., & Liu, H. (2014). Approaching human language with complex networks. *Physics of Life Reviews, 11*(4), 598–618.

Craven M., Freitag D., Mccallum A., & Mitchell T. (2003). Learning to extract symbolic knowledge from the World Wide Web, in *A Comprehensive Survey of Text Mining, M. W. Berry, Ed, Heidelberg, Germany: Springer*, 2003.

Devlin J., Chang M.W., Lee K. & Toutanova K.. (2019) "BERT: Pre-training of deep bidirectional transformers for language understanding", *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics Hum. Lang. Technol.*, vol. 1, pp. 4171-4186, 2019.

Foland W., & Martin J.H. (2017). Abstract meaning representation parsing using LSTM recurrent neural networks, In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL, 2017*, pp. 463–472, 2017.

Gao, H., Jennifer, Y., Li, S., Gu, M., Huang, Y., & Gong, B. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications, 73*(1), 220–239.

Garg, M., & Kumar, M. (2018). Identifying influential segments from word co-occurrence networks using AHP. *Cognitive Systems Research, 47*, 28–42.

Goh W.P., Luke K-K., & Cheong S.A. (2018). Functional shortcuts in language co-occurrence networks. *PLoS ONE*, 13 (9), e0203025. doi: 10.1371/journal. pone.0203025.

Grover A., & Leskovec J.N. (2016). node2vec: Scalable feature learning for networks, in: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855-864, Aug. 2016. doi: 10.1145/2939672.2939754.

Gupta, S., & Gupta, S. K. (2021). An approach to generate the bug report summaries using two-level feature extraction. *Expert Systems with Applications, 176*, Article 114816.

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin, 40*(3), 52–74.

Hassan, M. T., Karim, A., Kim, J.-B., & Jeon, M. (2015). Cdim: Document clustering by discrimination information maximization. *Information Sciences, 316*(20), 87–106.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 21*(9), 1263–1284.

Howard J. & Ruder S. (2018). Universal Language Model Fine-tuning for Text Classification, in: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, vol. 1, Jan. 2018.

Hu B., Lu Z., Li H., & Chen Q. (2014). Convolutional neural network architectures for matching natural language sentences, In: *Proceedings of the 27th Conference on Neural Information Processing Systems (NIPS'14)*, vol. 2, pp. 2042-2050, Dec. 2014.

Iglesias, E. L., Seara, V. A., & Borrajo, L. (2013). An HMM-based over-sampling technique to improve text classification. *Expert Systems with Applications, 20*, 7184–7192.

Junejo, K. N., Karim, A., Hassan, M. T., & Jeon, M. (2016). Terms-based discriminative information space for robust text classification. *Information Sciences, 372*, 518–538.

Kim Y. (2014). Convolutional neural networks for sentence classification, In: *Proceedings of EMNLP*, pp. 1746–1751, 2014.

Krawczyk, B. (2016). Learning from imbalanced data: Open challenges and future directions. *Progress in Artificial Intelligence, 5*, 221–232.

Le, Q., & Mikolov, T. (2014). *Distributed representations of sentences and documents* (vol. 32,, 1188–1196.

Li, Y., Guo, H., Zhang, Q., & Yang, J. (2018). Imbalanced text sentiment classification using universal and domain-specific knowledge. *Knowledge-Based Systems, 160*(15), 1–15.

Li, J., Zhu, J., & Zhang, B. (2016). Discriminative deep random walk for network classification. *ACL, 1*, 2016.

Li, Y., Wang, J., Wang, S., Liang, J. Y., & Li, J. Z. (2019). Local dense mixed region cutting + global rebalancing: A method for imbalanced text sentiment classification. *International Journal of Machine Learning and Cybernetics, 10*, 1805–1820.

Liang, H., Sun, X., Sun, Y. L., & Cao, Y. (2017). Text feature extraction based on deep learning: A review. *EURASIP Journal on Wireless Communications and Networking, 2017*, 211. https://doi.org/10.1186/s13638-017-0993-1

Ma, S. T., Xu, J., & Zhang, C. Z. (2018). Automatic identification of cited text spans: A multi-classifier approach over imbalanced dataset. *Scientometrics, 116*, 1303–1330.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *2013*. ICLR Workshop: Efficient estimation of word representations in vector space.

Naderalvojoud B., Akcapinar Sezer E., & Ucan A. (2015). Imbalanced Text Categorization Based on Positive and Negative Term Weighting Approach. in: *Proceedings of the 18th International Conference on Text, Speech, and Dialogue*, vol. 9302, pp. 325-333, Sep. 2015. doi: 10.1007/978-3-319-24033-6_37.

Ogura, H., Amano, H., & Kondo, M. (2011). Comparison of metrics for feature selection in imbalanced text classification. *Expert Systems with Applications, 38*(5), 4978–4989.

Pan, S., Wu, J., Zhu, X., Zhang, C., & Wang, Y. (2016). Tri-party deep network representation. *IJCAI, 2016*, 1895–1901.

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. *Empirical Methods in Natural Language Processing (EMNLP)*.

Perozzi B., AI-Rfou R., & Skiena S. (2014). DeepWalk: online learning of social representations, in: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701-710, Aug. 2014. doi:10.1145/2623330.2623732.

Perozzi B., Kulkarni V., & Skiena S. (2016). Walklets: multiscale graph embeddings for interpretable network classification, arXiv: 1605.02115 (2016).

Prihatini, PM, Suryawan, IK, & Mandia IN. (2018). Feature extraction for document using Latent Direchlet Allocation, *2nd International Joint Conference on Science and Technology (IJCST)*, SEP 27-28, 953, 012047. doi: 10.1088/1742-6596/953/1/012047.

Roshanfekr S., Esmaeili S., Ataeian H., & Amiri A. (2020). UGRWO-Sampling: A modified random walk under-sampling approach based on graphs to imbalanced data classification. arXiv preprint arXiv:2002.03521.

Shaikh, S., Daudpota, S. M., Imran, A. S., & Kastrati, Z. (2021). Towards improved classification accuracy on highly imbalanced text dataset using deep neural language models. *Applied Sciences, 11*, 869.

Song J., Huang X., Qin S., & Song Q. (2016). A bi-directional sampling based on K-means method for imbalance text classification. In: *IEEE/ACIS International Conference on Computer & Information Science IEEE*, Jun. 2016. doi: 10.1109/ICIS.2016.7550920.

Sun, Y., Wang, S. H., Li, Y. K., Feng, S. K., Tian, H., Wu, H., & Wang, H. F. (2020). ERNIE 2.0: A continual pre-training framework for language understanding. In *34th AAAI Conference on Artificial Intelligence* (pp. 8968–8975).

Tang, X., Mou, H., Liu, J. N., & Du, X. (2021). Research on automatic labeling of imbalanced texts of customer complaints based on text enhancement and layer-by-layer semantic matching. *Scientific Reports, 11*, 11849.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Comez A. N., Kaiser L. (2017). Attention is all you need, *31st Conference on Neural Information Processing (NIPS 2017)*, Long Beach, CA, USA. pp. 5999-6009.

Wang, S. G., Li, D. Y., Zhao, L. D., & Zhang, J. H. (2013). Sample cutting method for imbalanced text sentiment classification based on BRC. *Knowledge-Based System, 37*, 451–461.

Wu, Q. Y., Ye, Y. M., Zhang, H. J., Ng, M. K., & Ho, S. (2014). FORESTEXTER: An efficient random forest algorithm for imbalanced text categorization. *Knowledge-Based Systems, 67*, 105–116.

Xia, F., Wan, L. T., & Kong, X. J. (2020). Random walks: A review of algorithms and applications. *IEEE Transactions on Emerging Topics in Computational Intelligence, 4*(2), 95–107.

Xiao, Z., Wang, L., & Du, J. Y. (2019). Improving the performance of sentiment classification on imbalanced datasets with transfer learning. *IEEE Access, 7*, 28181–28290.

Yan, D. Y., Li, K. P., & Ye, J. (2019). Correlation analysis of short text based on network model. *Physica A: Statistical Mechanics and its Applications, 531*, Article 121728.

Yan, D. Y., Li, K. P., Gu, S., & Yang, L. (2020). Network-based bag-of-words model for text classification. *IEEE Access, 8*(1), 82641–82652.

Yang, Z., Tang, J., & Cohen, W. W. (2016). Multi-modal bayesian embeddings for learning social knowledge graphs. *IJCAI, 2016*, 2287–2293.

Yin, W. P., Schutze, H., Xiang, B., & Zhou, B. (2016). ABCNN: Attention based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics, 4*, 259–272.

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. https://doi.org/10.1109/MCI.2018.2840738

Zhang, H., & Li, M. F. (2014). RWO-Sampling: A random walk over-sampling approach to imbalanced data classification. *Information Fusion, 20*, 99–116.

Zhang X., Zhao J., & LeCun Y. (2015). Character-level convolutional networks for text classification, In: *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15)*, vol. 1, pp, 649-657, Dec. 2015.

Zhao, R., & Mao, K. Z. (2018). Fuzzy bag-of-words model for document representation. *IEEE Transactions on Fuzzy Systems, 26*(2), 794–804.

Zuo, Y., Zhao, J. C., & Xu, K. (2016). Word network topic model: A simple but general solution for short and imbalanced texts. *Knowledge and Information Systems, 48*, 379–398.