Detection of Stopwords in Classical Chinese Poetry

Lei Peng[1], Xiaodong Ma[2], Zheng Teng[3]
Library and Information Science Center, Chongqing Three Gorges Medical College, Chongqing, China[1]
Faculty of Data Science and Information Technology, INTI International University, Nilai, N. Sembilan, Malaysia[2]
School of International, Huanghe Science and Technology University, Zhengzhou, Henan, China[2]
School of Medical Technology, Chongqing Three Gorges Medical College, Chongqing, China[3]

Abstract—**In this research, we address the problem of stopword detection in Classical Chinese Poetry, an area that has not been explored previously. Stopword detection is crucial in text mining tasks, as identifying and removing stopwords is essential for improving the performance of various natural language processing models. Inspired by the TF-IDF method, we propose a novel approach that utilizes external knowledge to reconstruct the Term Weight matrix. Our key finding is that incorporating external knowledge significantly refines the granularity of the term weight, thereby improving the effectiveness of stopword detection. Based on these findings, we conclude that external knowledge can enhance the ability of text representation, especially for the short texts in Classical Chinese Poetry.**

Keywords—TF-IDF; stopwords; Chinese; poetry; frequency

I. Introduction

Stopwords are words that contain little semantic information and do not significantly contribute to text processing, despite their high frequency of occurrence [1, 2]. In text mining and information processing tasks (such as text classification and clustering), stopwords should generally be removed during the preprocessing stage [3, 4]. Removing stopwords can significantly improve the results of tasks such as feature extraction [5, 6], topic modeling [7], classification [8], ontology construction [9], and keyword extraction [10]. Stopwords have domain-specific characteristics, meaning that different domains have different stopword lists. They are typically a cluster of non-restrictive words. Since there is no fixed scope for stopwords, detecting them remains an evolving research field.

With the increasing popularity of classical Chinese poetry worldwide, more and more scholars are paying attention to it. Classical Chinese poetry is the pinnacle of Chinese traditional culture, and research on it will contribute to the development of Chinese culture. Currently, research in information retrieval and natural language processing in the Chinese language is mostly focused on modern Chinese, rather than classical Chinese. To the best of our knowledge, no researchers has yet conducted research on stopwords in classical Chinese poetry. Therefore, the significance of this study lies in our being the first to explore this area.

However, classical Chinese poetry is characterized by short texts, with many articles containing fewer than 20 tokens. These texts have low token repetition rates and are sparse, which makes them different from typical longer texts in text mining. Traditional methods, which rely solely on term frequency for stopword detection, face a challenge in this context, as they tend to result in nearly identical term frequencies for almost all terms. This leads to the issue of treating all terms equally. As a result, traditional methods often perform poorly when dealing with short texts. In this paper, we attempt to enhance the ability of text representation by using the TextRank method. We replace traditional Term Frequency with a finer measure of Term Importance, allowing for greater term index diversity in the text.

The structure of this paper is as follows: the first section introduces the research on stopwords in classical poetry; the second section reviews related research; the third section presents our proposed method; the fourth section describes our experiments, including the source of datasets, experimental processes, results, and discussion; finally, we conclude with a summary and outlook.

II. Related Works

Over the years, many researchers have explored stopwords issues, but we find that there has been little research on stopwords in Chinese classical poetry.

Zou et al. proposed a novel stopword list evaluation method using a mutual information-based Chinese segmentation approach [11]. Since this paper was published before the advent of Chinese automatic segmenters, its application scenario involved directly detecting stopwords in complete sentences. It uses mutual information values and sets thresholds and boundaries to calculate the association between grams.

Kucukyilmaz et al. used a classification approach to detect stopwords by constructing various features [12]. The features they used include frequency, term frequency, inverse document frequency, mean probability, variance probability, entropy, information model, and word positioning. They then evaluated their method using different classifiers.

Ferilli et al. employed Kullback-Leibler (KL) divergence as a measurement method and tested it on Italian language corpora [13]. This method is suitable for very small datasets, even those containing just one document.

Gerlach et al. used conditional entropy and a random null model to process stopwords [14]. Conditional entropy was used as the upper bound for the entropy of idealized stopwords, while the random null model was applied to compensate for undersampling. The difference between the two was then used as a criterion for measuring stopwords. The authors used quality assessment metrics such as NMI (Normalized Mutual

Information) for topic models and accuracy for classification tasks to experiment with the stopwords they detected.

Achsan et al. used the Term Frequency Inverse Document Frequency (TF-IDF) method to extract stopwords from a corpus collected from Indonesian online newspapers [15]. Since our method was inspired by their work, a detailed introduction to the method they used is provided here.

Specifically, $TF-IDF(t,d,D) = TF(t,d) \cdot IDF(t,D)$, where $TF-IDF(t,d,D)$ represents the TF-IDF score of a term *t* in the document *d* given the document collection *D*. For a vocabulary of size *V* and a total documents of number *M*, TF-IDF forms a *M*V* matrix, which has *M* rows (corresponding to the number of documents) and *V* columns (corresponding to the vocabulary size). Each entry in the matrix corresponds to a TF-IDF score of a specific term *t*.

$TF(t,d)$ refers to the frequency of a term *t* in the document *d*, calculated as the number of occurrences of the term in the document divided by the total number of terms in the document which is also the document length.

$$TF(t,d) = \frac{f(t,d)}{\sum_{t' \in d} f(t',d)} \tag{1}$$

in which, $f(t,d)$ is the number of times the term *t* appears in document d, and $\sum_{t' \in d} f(t',d)$ is the total number of occurrences of all terms in document *d*.

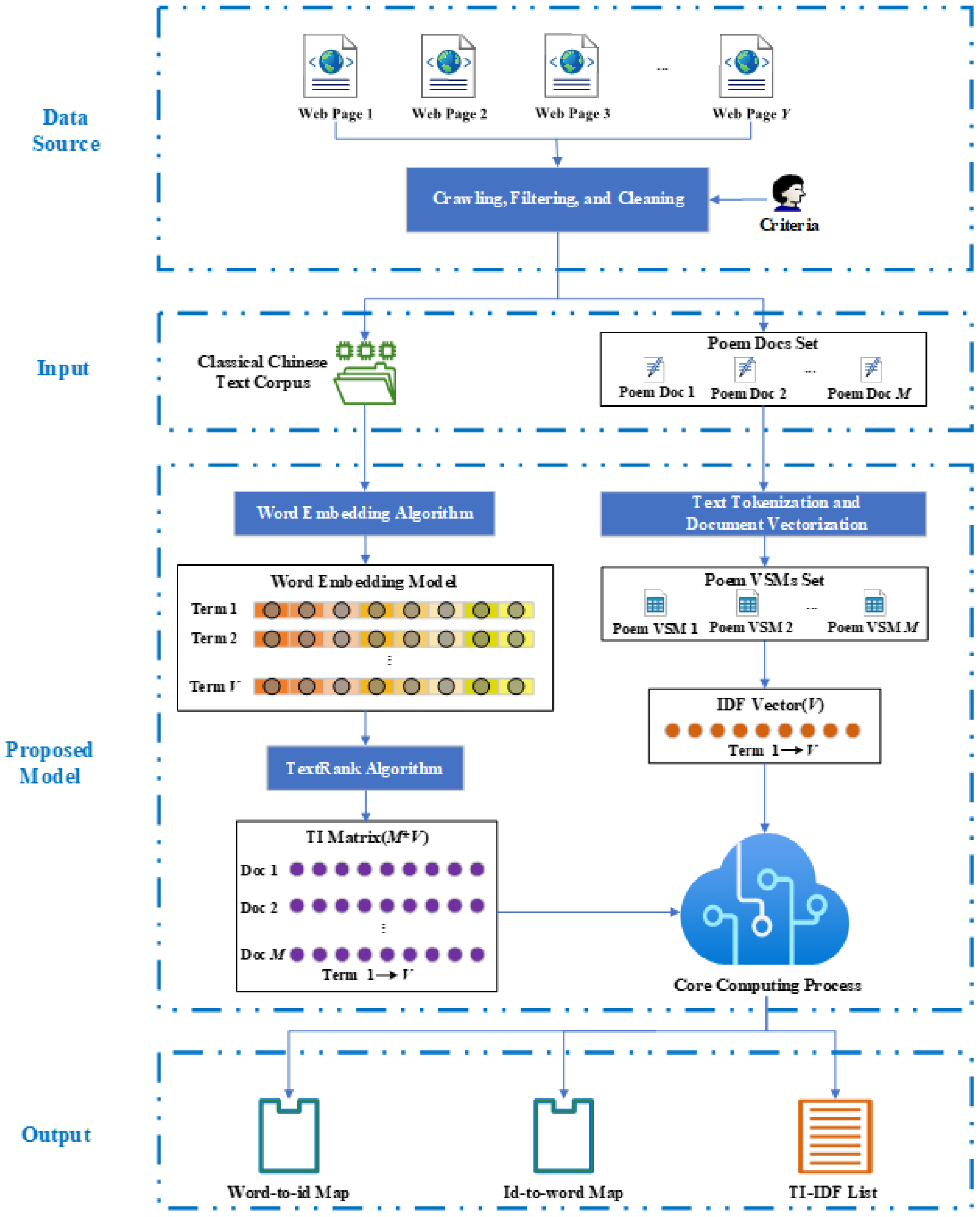


Fig. 1. Overall workflow comprising the proposed model.

$IDF(t, D)$ stands for Inverse Document Frequency of the documents comprising term *t* in total document collection *D*. The concept was introduced based on the idea that if a term *t* appears in most of the documents, it means the term does not help to distinguish between documents, and therefore it is not an important word. Its calculation method is the inverse operation of the proportion mentioned above with a logarithmic operation outside. It is a vector, as it is calculated for each term, making it a vector of length *V*.

$$IDF(t, D) = \log\left(\frac{N}{df(t)}\right) \quad (2)$$

in which, D is the document collection, N is the total number of documents in the document collection, and $df(t)$ is the number of documents that the word t appears in.

Finally, multiplying $TF(M, V)$ by $IDF(V)$ results in an *M*V* matrix. For each term's TF-IDF score, according to Achsan et al’s method, it should iterate through all documents, summing and averaging the TF-IDF values of the term. Then we list the average TF-IDF values of all terms in ascending order. Terms at the top are more likely to be stopwords, while terms at the bottom have a higher distinguishing ability for the document collection, meaning they are more likely to be significant terms.

Chinese language is considered a low-resource language, and research on stopwords in Chinese is still relatively scarce. Moreover, most available studies focus on modern Chinese, with seldom literature found on stopwords in Classical Chinese. The method proposed by Zou et al. is no longer applicable with the existence of Chinese word segmentation tools. Other studies primarily focus on long texts, which cannot be applied to short text corpora like Classical Chinese poetry. Kucukyilmaz et al. used a classification approach relying on various features such as word frequency and inverse document frequency. However, these features may not be effective for short texts with high contextual dependencies. Ferilli et al.'s use of the KL divergence method mainly measures the difference between two distributions, but for Classical Chinese poetry, which is structurally complex, lexically rich, and relatively short, KL divergence may not effectively capture the distributional characteristics. Gerlach et al.'s method relies on calculating conditional entropy, which is based on word frequency and may not effectively capture contextual relationships.

The Chinese poems are always short texts, while the all above methods are designed for long texts. Specifically, for the method used by Achsan et al., since each document researched in our research contains only a few terms, and each term appears only once, this leads to almost identical term frequency (TF) values for all terms in most documents. This creates significant challenges for the subsequent calculations.

III. Proposed Model

A. Overall Process

First, we crawl the required data from the Internet, which includes not only the classical Chinese poetry for this research but also the related external knowledge corpora. After obtaining all the data, we clean it according to the actual requirements. Once the data is cleaned, we divide it into two parts and feed them into the model.

On one hand, for the cleaned classical Chinese poetry, we perform character segmentation for each document and use Vector Space Model (VSM) to obtain vectorization. This is used to calculate the IDF vector for the entire poetry dataset. On the other hand, for the external knowledge, we perform word vector training and TextRank computation. This produces a Term Importance (TI) matrix, which we will explain in more detail in the next subsection. Afterward, we merge the TI and IDF to perform the Term Importance Inverse Document Frequency (TI-IDF) calculation. This is the core of our computation. Finally, we sort the resulting TI-IDF values in ascending order, and the terms on the top are more likely to be identified as stopwords. The overall workflow is shown in Fig. 1.

B. External Knowledge

- Step 1: We use word embedding technology to train the external knowledge corpus and obtain a word vector model. Word embedding is a technique that maps tokens from natural language into a real-valued space. It was first introduced by Bengio et al. in 2003 [16]. However, due to the complexity, it has not been given much attention until Mikolov et al. simplified the neural network architecture in 2013 [17]. After that Google implemented it and released the Word2Vec (W2V), which greatly advanced the development and application of word embedding technology. After training the word vector model, we can easily obtain the vector corresponding to each word, and then calculate the semantic similarity between words. The reason we obtain word vectors here is to perform the subsequent term importance (TI) computation in step 2.

- Step 2: The word vectors obtained in the previous step are then input into the TextRank algorithm. TextRank was proposed by Mihalcea et al. and inspired by Google’s foundational algorithm, PageRank [18]. The basic idea of PageRank is that if a page is referenced by many other pages, its importance is higher than that of others. TextRank applies this concept to text analysis. The computation unit is focused from web pages to sentences, and the link structure between web pages is replaced by the semantic similarity between sentences. The result is a ranking of sentence importance, and it is commonly used for tasks like key sentence identification and summary extraction. The TextRank formula is calculated as:

$$WS(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j) \quad (3)$$

in which, $WS(V_i)$ represents the weight of sentence *i*, and the sum on the right represents the contribution of each sentence to this sentence. In a single document, we can roughly think that all sentences are adjacent. W_{ji} represents the similarity of sentence j and *i*, and $WS(V_j)$ represents the weight of the last iterated sentence V_j . $In(V_i)$ means the precursor nodes of V_i , that the nodes point to V_i . $Out(V_j)$

means the follow-up nodes of V_j, that the nodes point out from V_j.

In our model, we adjust the TextRank method by changing the computation unit from a sentence to a token. Using the word vectors from the previous step, we calculate the semantic similarity between terms. We then input the constructed token similarity matrix into TextRank, ultimately obtaining the term importance matrix. This process is described in Algorithm 1.

Algorithm 1: Compute term importance
Input: The word embedding pretrained model *wv*: {term: term_vec}, the document index *d*.
Output: The term importance dict, the corresponding term count dict.

```
# term count
token_id_list[] = Document[$d$].get_tokens()
term_count_dict = {}
for token_id in token_id_list:
    if token not in term_count_dict:
        term_count_dict[token_id] = 1
    else:
        term_count_dict[token_id] += 1
# compute distinguished terms list
term_id_list[] = term_count_dict.keys()
# compute TextrRank value
similarity = [][]
    for term_1 in term_id_list:
        for term_2 in term_id_list:
            similarity[term_1][term_2] = compute_similarity(wv(term_1),wv(term_2))
term_TR_dict{} = TextrRank(similarity)
return term_TR_dict, term_count_dict
```

C. Core Computation

For the obtained term_TR_dict (term TextRank dictionary) and term_count_dict (term count dictionary), we will arrange the keys according to the vocabulary order in the VSM to get the term importance vector $\overrightarrow{\zeta_d} = [\zeta_{d\_1}, \zeta_{d\_2}, \cdots, \zeta_{d\_v}]$ and the corresponding term count vector $\overrightarrow{\lambda_d} = [\lambda_{d\_1}, \lambda_{d\_2}, \cdots, \lambda_{d\_v}]$. Here, d_i represents the i-th term in the vocabulary of document d. ζ_{term} and λ_{term} represent the importance value and count value corresponding to the term. We multiply $\overrightarrow{\zeta_d}$ with $\overrightarrow{\lambda_d}$, and then apply softmax to obtain the term importance result. It is important to note that the softmax operation is used to smooth the data and ensure that all values fall within the range of 0 to 1.

$$Term\ Importance(d\_i) = softmax(\zeta_{d\_i} \cdot \lambda_{d\_i}) = \frac{e^{\zeta_{d\_i} \cdot \lambda_{d\_i}}}{\sum_{j=1}^{d_v} e^{\zeta_{d\_j} \cdot \lambda_{d\_j}}}\ for\ i = 1,2,\cdots,d_v \quad (4)$$

After obtaining the term importance list, its length will be d_v, which is the number of terms in the document, d's vocabulary, rather than the length of the entire vocabulary V, D's vocabulary. We should map it to the entire vocabulary to obtain the vector, which will be constructed as row in the Term Importance matrix. For terms that do not appear in the document, we set their values to 0. Below is the pseudocode for this process:

```
z_d_final[] = zero($V$)
for $i$ in [1, $V$]:
    z_d_final[$i$] = Term Importance[vocabulary[$i$]]
return z_d_final
```

This is for just one document, and we need to perform this operation on all documents to ultimately obtain the entire term importance matrix. For IDF, we still use the calculation method from the TF-IDF approach. Therefore, the final calculation formula is:

$$TI - IDF(t, d, D) = TI(t, d) \cdot IDF(t, D) \quad (5)$$

in which:

$$TI(t, d) = Term\ Importance(t)\ on\ Doc[d] \quad (6)$$

and

$$IDF(t, D) = \log\left(\frac{N}{df(t)}\right) \quad (7)$$

For all terms, we also apply the method used in Achsan's paper, computing the average to obtain the final result.

$$TI - IDF(t, D) = \frac{\sum_{d=1}^{N} TI(t,d)}{df(t)} \cdot \log\left(\frac{N}{df(t)}\right) \quad (8)$$

IV. Experiment

A. Dataset

The Tang poetry and Song poetry, with one representing the highest quality and the other the largest quantity, are suitable to be the subjects in this research. Since these two datasets are publicly available on the Internet, we crawled them from online sources[1, 2]. Finally, we obtain Complete Tang Poems (CTP) dataset and Poems of Song Dynasty (PSD) dataset. As the main focus of this paper is on stopword detection, the details of the crawling, storing, tokenizing, and cleaning process are not elaborated here due to space limitations.

TABLE I. DATASETS DETAILS

-	CTP Dataset	PSD Dataset
Document #	42,479	182,213
Vocabulary Size	7,062	11,230
Min length	3	5
Max length	3,750	2,013
Avg length	58.62	58.71

To verify the efficiency of our stopword detection, we need to set some stopwords as criteria. We invited three graduate students majoring in Chinese language and literature to each list some stopwords based on their knowledge of poetry. Then, we extracted the common words from the three lists, which totals 54 words. Finally, this stopwords list have received

[1] http://www.wenxue100.com/book_GuDianShiCiWen/5.thtml
[2] http://www.wenxue100.com/book_GuDianShiCiWen/26.thtml

unanimous approval from the three volunteers, and we published it here.[3]

B. External Information

We downloaded a dataset used as studying classical Chinese texts for ancient Chinese people from the website4. The dataset includes Confucian classics (Jing), History (Shi), Philosophy (Zi), and Literature (Ji). It contains 43 million Chinese characters and a vocabulary length of 16,413, making it well-suited for training word embedding model.

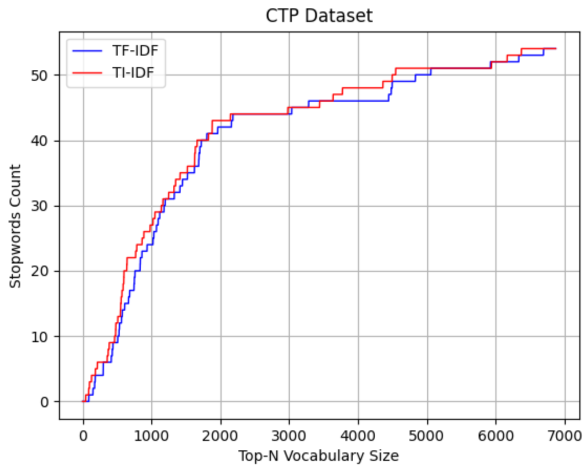
C. Running Configuration

Our program runs in the following environment: a PC with Windows 10, an Intel Xeon E5-2680 CPU (2.4GHz 2 Cores), 64GB of RAM, and a 1TB hard drive. The programming language is Python 3.7, with the development environment PyCharm 2021.3 and Anaconda 4.7.10. For word embeddings, we use Gensim 4.2.0, with parameters set to CBOW and Negative Sampling, and a word vector dimension of 100. For TextRank, we use NetworkX 2.6.3 for the implementation, and the damping factor d is set to 0.85, as in most other researches.

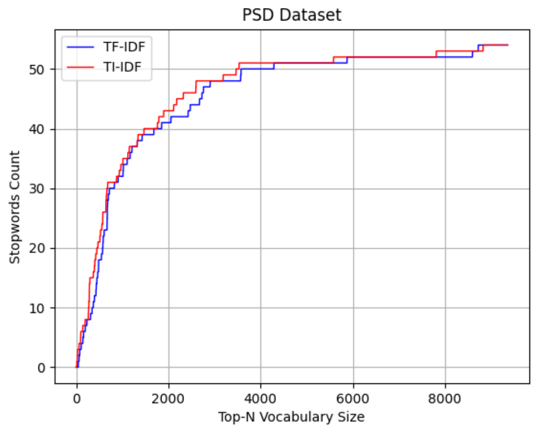
D. Results

We can observe that the trend of our model is similar to that of the original model shown in Fig. 2. As shown in Fig. 2, in nearly all regions, the number of stopwords detected by our method exceeds that of the TF-IDF method, except for a few specific intervals, such as in Fig. 2(a) when the TOP-N range is approximately within [1750, 1800] and [3300, 3400], and in Fig. 2(b) when the TOP-N range is approximately within [6900, 7000].

In Fig. 2(a), when TOP-N is in the range of 0 to 2000, the number of stopwords increases rapidly. After TOP-N exceeds 2000, the growth rate of stopword numbers levels off. In Fig. 2(b), when TOP-N is in the range of 0 to 1000, the number of stopwords significantly increases, then slows down between 1000 and 3500, and eventually experiences a smooth upward trend after TOP-N exceeds 3500.

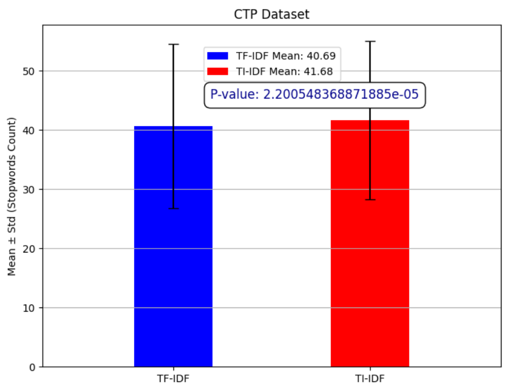


(a) Stopwords count details on CTP dataset.

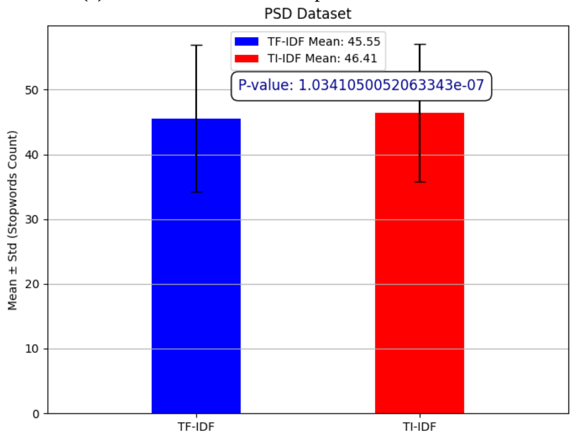


(b) Stopwords count details on PSD dataset.

Fig. 2. Stopwords count comprised in top-n terms.



(a). Statistical features comparison on CTP dataset.



(b). Statistical features comparison on PSD dataset.

Fig. 3. Statistical features comparison of the results.

Additionally, we compared our results with those of the TF-IDF method, using these two sets of data for analysis through the mean, standard deviation, and *t*-test. The results are shown in Fig. 3, where Fig. 3(a) shows the comparison on the CTP dataset and Fig. 3(b) shows the comparison on the PSD dataset. We found that the mean of our method exceeds that of the TF-IDF method by approximately 0.9 percentage points across both datasets. In terms of standard deviation, our method generally has a smaller standard deviation compared to the TF-IDF method, except in the CTP dataset, where the

[3] https://github.com/powerwings0377/stwords_list
[4] https://so.gushiwen.cn/guwen

standard deviation of the TI-IDF method is slightly higher than that of the TF-IDF method. The *p*-values for both datasets in *t*-test are far less than 0.01, indicating that the results of our method show statistical significance in the comparison experiment.

E. Discussion

Our model exhibits similar performance across the two corpora. For example, when TOP-N is 1000, the stopwords count for both corpora is approximately 30, and when TOP-N is 2000, the stopwords count is around 40. However, there are differences in performance between the two corpora. For instance, the rate of increase in stopwords count differs due to the size of the vocabulary. The CTP corpus has a vocabulary size of around 6000, while the PSD corpus has a vocabulary size of over 11,000. The vocabulary in the CTP corpus is more condensed, which causes a steeper increase in stopwords count. When TOP-N reaches 4000, the stopwords count in the PSD corpus exceeds 50 and starts to level off, while in the CTP corpus, the stopwords count is still rising. Additionally, as shown in Fig. 3, we can conclude that as the number of documents and the vocabulary size increase, the detection of stopwords improves.

The reason our method outperforms the original method in most areas is that we incorporate external knowledge into the term weight calculation, which introduces a "preference" factor into the weight computation. In the original method, the weight is computed based on term frequency, which leads to equal weights for equal frequencies. Since most terms in a poetry document appear only once, terms, whether important or unimportant, are treated the same. In our method, important terms are assigned higher weights, making previously equal weights become fine-grained. This increases the term index diversity and makes the calculation results more rational.

V. Conclusion

As a low-resource language, classical Chinese desires information technology processing all the time. As the detection of stopwords in Chinese classical poetry has never been researched before, in this paper, we proposed a TI-IDF method to address this issue, in which large-scale classical Chinese resources are used as an external knowledge base. By utilizing word embeddings and TextRank, we constructed a Term Importance matrix to replace the Term Frequency matrix in the original TF-IDF method. We found that the Term Importance matrix constructed in this paper provided a more refined calculation of term weights compared to the Term Frequency matrix, as external knowledge plays a key role in fine-tuning the process. Our excellent performance on the CTP and PSD datasets also validates the reliability of our method.

This paper explores stopword detection in classical Chinese poetry. The effectiveness of our proposed method largely depends on the selection of external knowledge. Due to the use of a pre-trained word embedding model and the construction of the TextRank network, there was an increase in training time; however, this does not affect the improvement in model performance. For future work, we aim to explore labeled datasets and conduct research on the impact of text analysis performance with stopword removal.

Acknowledgment

This work was supported by the Chongqing Three Gorges Medical College of China (No. 2019XZYB13) and by the Chongqing Association of Higher Education under Chongqing Municipal of China (No. CQGJ21B128).

References

[1] J. Kaur and P. K. Buttar, "A systematic review on stopword removal algorithms," International Journal on Future Revolution in Computer Science & Communication Engineering, vol. 4, no. 4, pp. 207-210, 2018.

[2] M. Dehghani and M. Manthouri, "Semi-automatic detection of Persian stopwords using FastText library," in 9781665402088, 2021.

[3] S. Sahu and S. Pal, "Effect of stopwords in Indian language IR," Sadhana - Academy Proceedings in Engineering Sciences, vol. 47, no. 1, pp. -, 2022.

[4] A. Bichi, R. Samsudin and R. Hassan, "Automatic construction of generic stop words list for hausa text," Indonesian Journal of Electrical Engineering and Computer Science, vol. 25, no. 3, pp. 1501-1507, 2022.

[5] R. Arlitt, S. Khan and L. Blessing, "Feature engineering for design thinking assessment," in International Conference on Engineering Design, 2019.

[6] K. Goucher-Lambert and J. Cagan, "Crowdsourcing inspiration: using crowd generated inspirational stimuli to support designer ideation," Design Studies, vol. 61, pp. 1-29, 2019.

[7] H. Song, J. Evans and K. Fu, "An exploration-based approach to computationally supported design-by-analogy using D3," AI EDAM, vol. 34, pp. 444-457, 2020.

[8] S. Urologin, "Sentiment analysis, visualization and classification of summarized news articles: a novel approach," (IJACSA) International Journal of Advanced Computer Science and Applications,, vol. 9, no. 8, pp. 616-625, 2018.

[9] F. Shi, L. Chen, J. Han and P. Childs, "A data-driven text mining and semantic network analysis for design information retrieval," Journal of Mechanical Design, vol. 139, no. 11, 2017.

[10] B. Guda, B. K. Nuhu, J. Agajo and I. Aliyu, "Performance evaluation of keyword extraction techniques and stop word lists on speech-to-text corpus," International Arab Journal of Information Technology, vol. 20, no. 1, pp. 134-140, 2023.

[11] F. Zou, F. L. Wang, X. Deng, and S. Han, "Evaluation of Stop Word Lists in Chinese Language," in Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006, Genoa, Italy, May 22-28, 2006, pp. 2497-2500.

[12] T. Kucukyilmaz and T. Akin, "A Feature-based Approach on Automatic Stopword Detection," in Intelligent Systems and Applications, K. Arai, Ed., Lecture Notes in Networks and Systems, vol. 825, Springer, Cham, 2024.

[13] S. Ferilli, G. L. Izzi, and T. Franza, "Automatic Stopwords Identification from Very Small Corpora," in Intelligent Systems in Industrial Applications, M. Stettinger, G. Leitner, A. Felfernig, and Z. W. Ras, Eds., Studies in Computational Intelligence, vol. 949, Springer, Cham, 2021.

[14] M. Gerlach, H. Shi, and L. A. N. Amaral, "A universal information theoretic approach to the identification of stopwords," Nat Mach Intell, vol. 1, pp. 606–612, 2019. https://doi.org/10.1038/s42256-019-0112-6.

[15] H. T. Yani Achsan, H. Suhartanto, W. C. Wibowo, D. A. Dewi, and K. Ismed, "Automatic Extraction of Indonesian Stopwords," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 14, no. 2, 2023.

[16] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," Journal of Machine Learning Research, vol. 3, pp. 1137-1155, 2003.

[17] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," Proceedings of the International Conference on Learning Representations (ICLR 2013). Available: http://arxiv.org/abs/1301.3781.

[18] R. Mihalcea and P. Tarau, "TextRank: Bringing order into texts," Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004), pp. 404-411. Available: https://www.aclweb.org/anthology/W04-3252/.