**Implementation and improvement of text retrieval algorithm based on TF-IDF model**

Li Zhiyu Li Xiang

South China Normal University South China Normal University

20214001007@m.scnu.edu.cn 20214001028@m.scnu.edu.cn

[firstauthor@i1.org](mailto:firstauthor@i1.org) [secondauthor@i2.org](mailto:secondauthor@i2.org)

Luo Zijie Wu Jiaen

South China Normal University South China Normal University

20214001084@m.scnu.edu.cn 20214001075@m.scnu.edu.cn

[thirdauthor@i1.org](mailto:firstauthor@i1.org) [fourthauthor@i2.org](mailto:secondauthor@i2.org)

**Abstract**

*TF-IDF (Term Frequency-Inverse Document Frequency) is a query algorithm that is widely used in text retrieval. However, the fitting effect of the traditional TF-IDF algorithm still has shortcomings. In this experiment, we abandoned the method of using the polynomial naïve Bayes classifier MultinomialNB to train the model, and replaced the neural network model LSTM and BERT to experiment on the dataset. In fact, our experimental results show that the BERT model works best on our dataset. However, the TF-IDF model is not as good as the neural network model in terms of fitting effect, so it can be concluded that our method is effective.*

# **Introduction**

**1.1Motivation and problem solving scheme**

In the current digital information age, the rapid growth of news data has made efficient classification of these massive amounts of information an urgent need in the field of information processing. Revealing the characteristics and differences of texts from different news sources is crucial for gaining a deeper understanding of information and making targeted analysis. Therefore, the motivation of the research is to meet the growing demand for information classification and provide solutions for more effective processing and understanding of news data. In order to achieve this goal, we abandoned the early use of Bayesian naive classification schemes and instead adopted neural network models to improve the TF-IDF model. We conducted comparative experiments using RNN (LSTM), BERT, and the original TF-IDF model, and improved them to find the most effective solution to solve the target task.

**1.2Summary of innovation points**

We compared the original TF-IDF model with its LSTM and BERT models, and compared its performance differences on different datasets (long/short text) and whether the dataset is simple or not. From this, we obtained the best solution for improving TF-IDF for text classification tasks. In the design of the LSTM model, we adopted bidirectional (forward and backward) LSTM layers to obtain the maximum available information. The output of this layer (pushed to the next layer) is the output of the last word (right for forward LSTM, left for backward LSTM). In terms of the BERT model, we used the Distill Bert model with a small number of parameters, as well as the Python library converter provided by HuggingFace (together with many other models).

# **Related work**

**2.1Research Questions**

The research on text retrieval based on the TF-IDF model primarily focuses on designing and constructing efficient methods to retrieve text data on specific topics from a large corpus. In practical applications, users often need to sift through a large volume of textual data to find the information they seek. However, these datasets often contain significant amounts of noise and redundant

information. Therefore, extracting useful and relevant information accurately from such data is a crucial problem.

In general, the main aspects of the problem include the following:

1. To design an effective feature extraction method that can comprehensively describe the information in raw text data and reduce noise.

2. To design a feature that takes into account information characteristics such as document length, lexical diversity, and semantics to improve model accuracy and robustness.

3. To effectively handle challenges such as long texts, difficult-to-process texts, and semi-structured texts to improve the model's processing capability and efficiency.

4. To design an effective query model that accurately understands and matches the relationship between query terms and documents, improving query accuracy.

5.To evaluate the performance of a model and improve its reliability and practicality.

**2.2Subjects and objectives of the study**

**2.2.1Subjects of the study**

Due to the wide applicability of TF-IDF-based text retrieval, the research objects can be any entities that require text information retrieval, such as websites, blogs, search engines, applications, input methods, and more

**2.2.2objectives of the study**

1. Assisting users in quickly and accurately retrieving relevant textual information: By utilizing text data generated using the TF-IDF algorithm, users can retrieve documents or information related to specific keywords or phrases through simple query statements

2. Improving the performance of a search engine: The TF-IDF algorithm can efficiently calculate the weights of keywords or phrases in text data, allowing for better indexing and ranking of documents by search engines. This improves search efficiency.

3.Support for unstructured text data: The TF-IDF algorithm can handle unstructured text data, such as text files and web content, without requiring manual annotation of these data.

4.Scalability: The TF-IDF algorithm can be easily applied to large amounts of text data, enabling the implementation of large-scale text retrieval services effortlessly.

**2.2.3technologies**

Over the years, TF-IDF-based text retrieval methods

can generally be classified into two categories: statistical methods and machine learning methods.

Statistical methods primarily calculate the importance

of words in the text using statistical measures of TF-IDF scores, such as mean, median, variance, and standard deviation. Preprocessing the initial text data with resources like WordNet can provide better text statistics, thereby improving the accuracy and effectiveness of the TF-IDF model (Leena H. Patil, 2013). In tasks like efficient text label matching frameworks for extreme multi-label text classification, using MatchXML can effectively enhance the precision and speed of the TF-IDF model, achieving the goal of algorithm improvement (Hui Ye, 2023).

Machine learning methods, on the other hand, leverage deep learning models to calculate the importance of words in the text. Common models include Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) networks, Support Vector Machines (SVM), Decision Trees (DT), and more. For example, in the paper "ALJP: An Arabic Legal Judgment Prediction in Personal Status Cases Using Machine Learning Models," SVM, Logistic Regression (LR), LSTM, and Bidirectional LSTM (BiLSTM) were tested on TF-IDF and word2vec representations, yielding conclusive results.(Salwa Abbara,2023).

**3.Dataset\_description**

A new dataset was applied in this experiment: the BBC News(https://www.kaggle.com/sainijagjit/bbc-dataset).The BBC News Dataset is a common dataset used for text classification tasks, covering text information from various fields such as sports, politics, technology, and art, including the following two attributes:

1. Category: Each text sample has a corresponding label, indicating the category to which the text belongs. Such as "Politics", "Sports", etc.

2. Text (Text Content): Text data contains the main text, title, or other relevant information from news articles.

Due to the repeated validation and use of the dataset in various existing experiments, no improvements were made to the dataset in this experiment.)

Data examples can be found in the appendix.

**4.Approach**

**4.1Model Architecture**

Since our TFIDF model uses the original model, we will not repeat it here.

**4.1.1 LSTM**

Our LSTM model is constructed as follows:

Input layer: to tell the model which input format to expect, so that the model knows what to expect

Embedding: we transform the input (a sequence of word indices) into a sequence of embedded words (a sequence of vectors of size 300), using the downloaded Word2Vec matrix

LSTM layer: we use an LSTM layer that goes in both directions (forward and backward), to have maximal information available. The output of this layer (that is pushed to the next layer) is the output of the last word (on the right for the forward LSTM, on the left for the backward LSTM). We set the size of the output vector to 15 (which is somewhat arbitrary). Combining both outputs (forward and backward), we get a vector of size 30

Dropout layer: for regularization

Dense layer (with relu activation function, with 64 neurons): to solve the specific problem of classification

Dense layer (with softmax activation function): for a probability distribution for each label

We run the model using Adam optimizer, where we have played with the hyper-parameters. Based on the recommendations of the first article on dropout (2014) [7], we increase the learning rate compared to TensorFlow's default values.

**4.1.2 BERT**

The BERT consists of:

Input layer: to tell the model which input format to expect, so that the model knows what to expect

Distil Bert model: to encode the input data into a new sequence of vectors (that is the output of BERT). Only the first vector of this sequence will be used as an input for the rest of the classifier

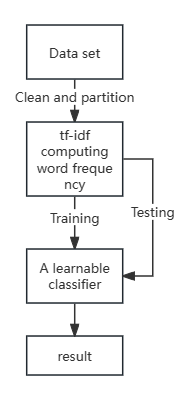
Dropout layer: for regularization

Dense layer (with relu activation function, with 64 neurons): to solve the specific problem of classification

Dense layer (with softmax activation function): for a probability distribution for each label

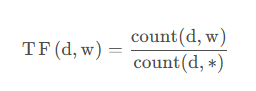
**4.2 Model formulas**

The schematic diagram of our model is roughly as follows:



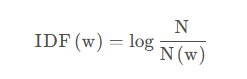
The learnable classifiers are LSTM and BERT

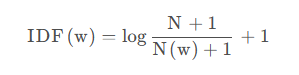
TF-IDF formula:



count(d, w): The number of times the word w appears in document d.

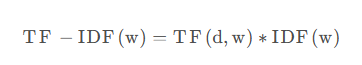
count(d, \*): The total number of words in document d.





N: The total number of documents in the corpus.

N(w): How many documents in which the word w appears.



**4.3Optimize the goal and loss function**

In the experiment, we adopted the following definitions of loss functions:

LSTM loss function:

opt = optimizers.Adam(learning\_rate=0.01, beta\_1=0.9)

model.compile(optimizer=opt,loss='categorical\_crossentropy', metrics=['accuracy'])

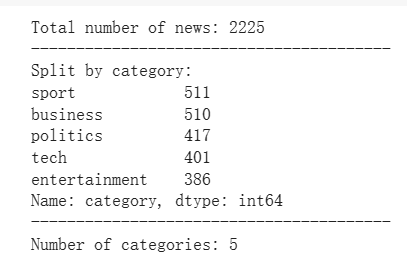
BERT loss function:

dmodel.compile(optimizer='adam',loss='categorical\_crossentropy', metrics=['accuracy'])

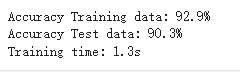
**5.Expermental results**

**5.1data classification**

After classifying the data, we obtained a total of 2,225 entries, which were relatively evenly distributed across the five categories.



**5.2The test results of TF-IDF**

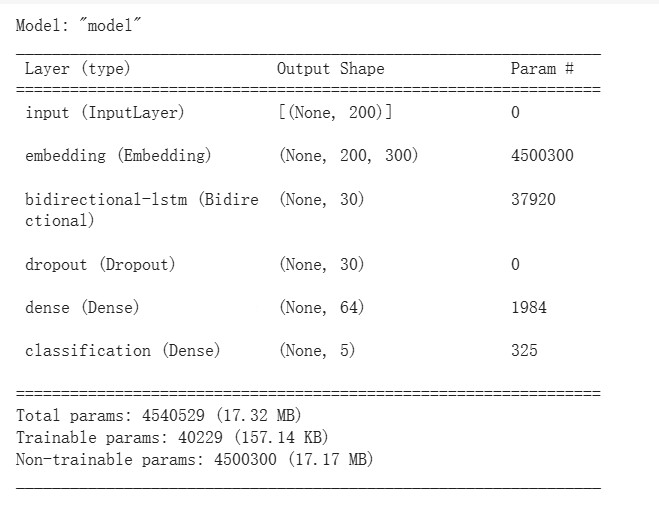


It can be seen that the accuracy of the training set of the model is 92.9%, the accuracy of the test set is 90.3%, and the training time is 1.3s, indicating that it has good generalization ability and fast processing results.

**5.3The test results of LSTM**

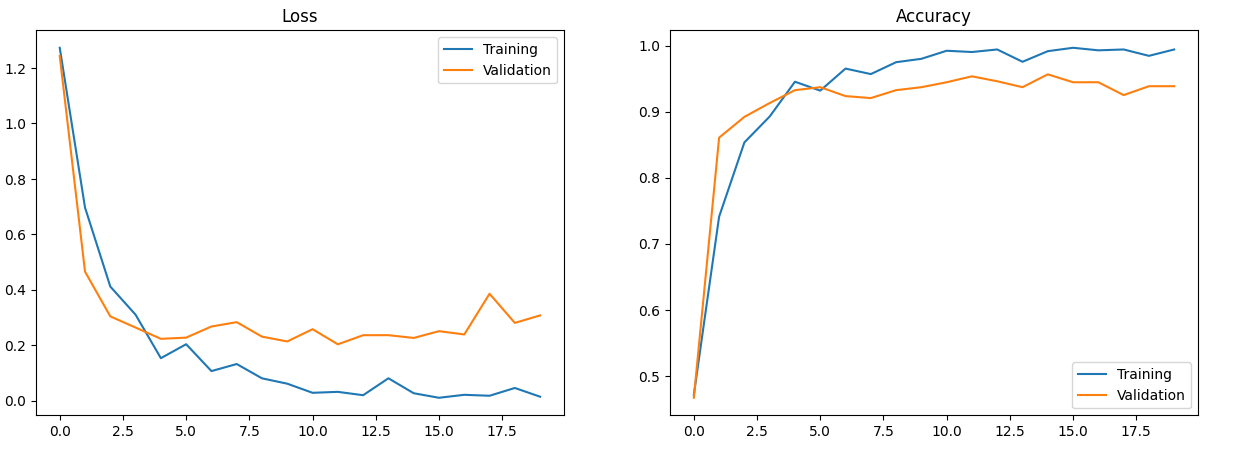
And then we used a LSTM model, using word embeddings to take advantage of pre-learned information.

The results of the experiments obtained



8288d208f228327b1455534bb2528a6

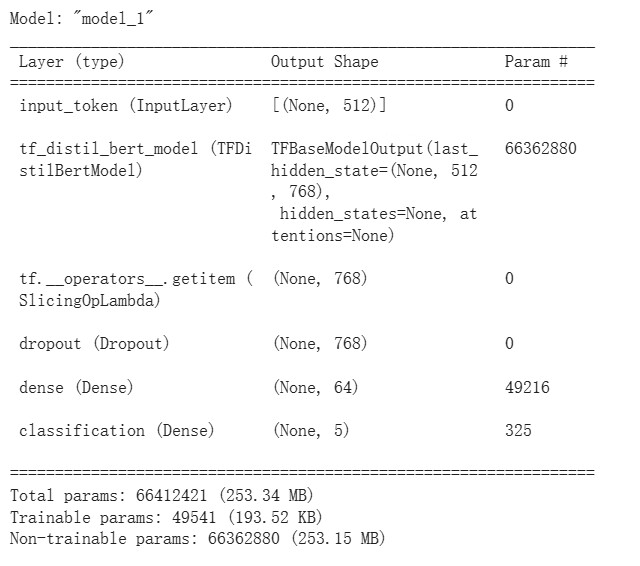
We can see that the accuracy of the training set of the model is 98.2%, the accuracy of the test set is 94.2%, and the training time is 13.5s, for this (simple) problem with little training data, we observe that this complex model is more accurate than the TF-IDF model, but it takes about 10 times the training time, which may be disappointing. The recurrent neural network model that makes Word2Vec is an improvement on the linear TF-IDF model, which takes into account the order of words and uses pre-learned word representations. However, we do not take into account the context in which the word is spoken. We think it's reasonable that context matters. This is the problem that the BERT model is solving.

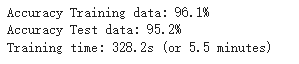


loss accuracy

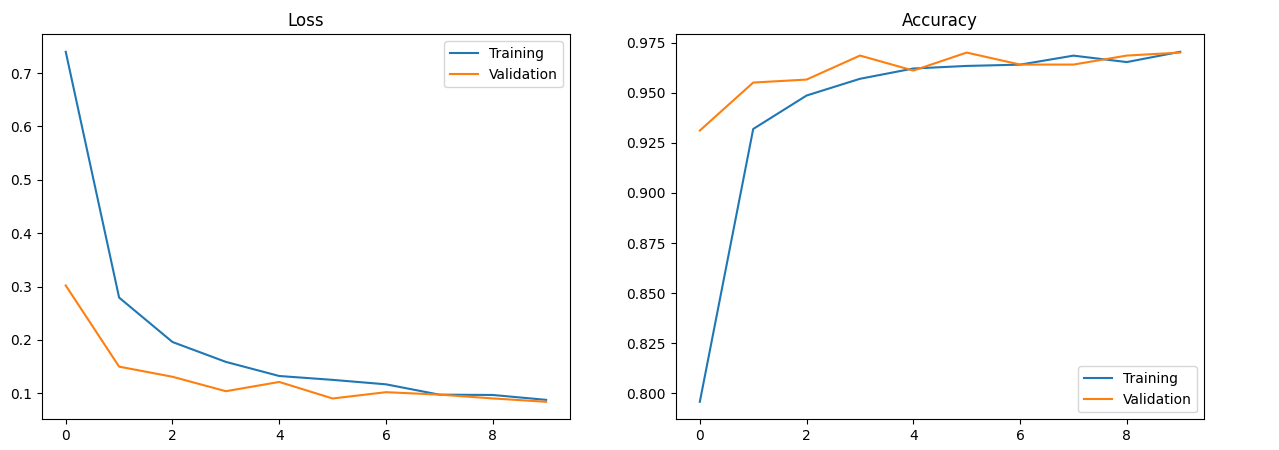
**5.4The test results of BERT**

Finally, we used a BERT model, which is a pre-trained model that is pre-trained not only on individual words, but also on entire sentences**.**



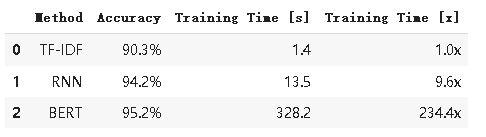


It can be seen that the accuracy of the training set of the model is 96.1%, the accuracy of the test set is 95.2%, and the training time is 328.2s. For this (simple) problem with little training data, we observed that this complex model was more accurate than the TF-IDF model, but took more than 200 times longer to train.



loss accuracy

**5.5 Experimental analysis**



General table

From the overall data, we can make the following analysis:

In the case of low data volume:If the amount of data is small, you may find that the first model performs better with a short training time because it has a relatively short training time and is relatively accurate.

Medium data volume and computing resources:The second model strikes a balance between accuracy and training efficiency, and is suitable for medium-sized datasets and computing resources. It performs relatively well in terms of accuracy, and the training time is within acceptable limits.

High accuracy requirements and sufficient computing resources: If there are higher requirements for accuracy and there are sufficient computing resources, then the third model may be a good choice. It performs well in accuracy but takes longer to train.

Real-time requirements:If your app requires real-time performance, you may want to consider a model with a shorter training time, i.e., the first or second model

**6.Team member contribution**

Luo Zijie: Part of the paper writing, code running (20%)

Li Xiang :Code Improvement, Dataset Development, code running (30%)

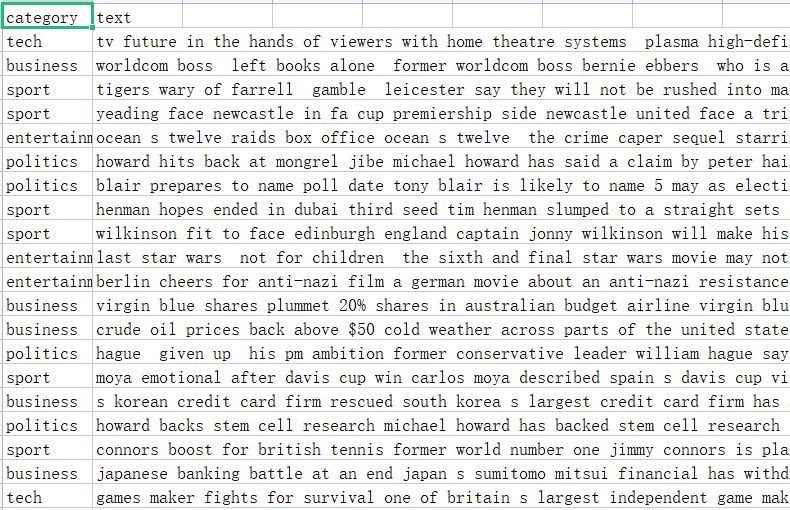
Wu Jiaen: Part of the paper writing，Code Improvement (20%)

Li Zhiyu: Part of the paper writing, code improvement and running(30%)

**7.Acknowledgement**

No body but ourselves.Thanks for our teamates’ hard work.

**Appendix**

Data examples

# **References**

1. Leena H. Patil, Mohammed Atique. A Semantic approach for effective document clustering using WordNet. *arXiv:1303.0489 [cs.CL]*.
2. Hui Ye, Rajshekhar Sunderraman, Shihao Ji. MatchXML: An Efficient Text-label Matching Framework for Extreme Multi-label Text Classification. *arXiv:2308.13139 [cs.CL]*.
3. Salwa Abbara, Mona Hafez, Aya Kazzaz, Areej Alhothali, Alhanouf Alsolami. ALJP: An Arabic Legal Judgment Prediction in Personal Status Cases Using Machine Learning Models. *arXiv:2309.00238 [cs.AI].*
4. Bakhyt Bakiyev.Method for Determining the Similarity of Text Documents for the Kazakh language, Taking Into Account Synonyms: Extension to TF-IDF. *arXiv:2211.12364 [cs.IR]*
5. Amir Jalilifard, Vinicius F. Caridá, Alex F. Mansano, Rogers S. Cristo, Felipe Penhorate C. da Fonseca. Semantic Sensitive TF-IDF to Determine Word Relevance in Documents. arXiv:2001.09896 [cs.IR]
6. Mikolov, Tomas, et al. (2013). "Efficient Estimation of Word Representations in Vector Space"
7. Srivastava, Hinton et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting"
8. Devlin et al. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"
9. 邸剑,刘骏华,曹锦纲.利用BERT和覆盖率机制改进的HiNT文本检索模型[J/OL].智能系统学报,1-8[2024-01-13]http://kns.cnki.net/kcms/detail/23.1538.TP.20230926.1452.002.html.
10. 高士杰. 基于BERT和文本分割的上下文文本检索技术研究[D]. 华中科技大学, 2020. DOI:10.27157/d.cnki.ghzku.2020.002192
11. 吴远云.基于改进TFIDF和LSTM的中文文本分类算法研究[D].江西财经大学,2023.DOI:10.27175/d.cnki.gjxcu.2023.001265