applied sciences

MDPI

Article

Improving Performance of Massive Text Real-Time Classification for Document Confidentiality Management

Lingling Tan *, Junkai Yi and Fei Yang

Institute of Automation, Beijing Information Science and Technology University, Beijing 100192, China; yijk@bistu.edu.cn (J.Y.); yangfei@bistu.edu.cn (F.Y.)
* Correspondence: tanlingling@bistu.edu.cn

Abstract: For classified and sensitive electronic documents within the scope of enterprises and organizations, in order to standardize and strengthen the confidentiality management of enterprises and meet the actual needs of secret text classification, a document automatic classification optimization method based on keyword retrieval and the kNN classification algorithm is proposed. The method supports keyword classification management, provides users with keywords of multiple risk levels, and then combines a matching scanning algorithm to label keywords of different levels. The text with labels is used as the training set of the kNN algorithm to classify the target text and realize the classification protection of text data. Aimed at solving the shortcomings of large feature vector dimension, low classification efficiency, and low accuracy in existing kNN text classification methods, an optimization method is proposed using a feature selection algorithm and a kNN algorithm based on an AVX instruction set to realize real-time classification of massive texts. By constructing a keyword dictionary and an optimized feature vector, parallel calculation of the feature vector weight and distance vector is realized, and the accuracy and efficiency of text classification are improved. The experimental results show that the multi-classification effect of the feature selection algorithm used in this paper, *tf-DE*, is better than that of the traditional *tf-idf* algorithm, and the classification effect of kNN is comparable to that of the support vector machine (SVM) algorithm. With the increase in feature vector dimensions, the classification effect of the text classification algorithm is improved and the classification time also increases linearly. The AVX-256 acceleration method takes about 55% of the time of the original version, thus verifying the effect of multi-classification of massive texts for document confidentiality management.

Keywords: kNN; text classification; AVX-256; feature vector

Citation: Tan, L.; Yi, J.; Yang, F. Improving Performance of Massive Text Real-Time Classification for Document Confidentiality Management. *Appl. Sci.* **2024**, *14*, 1565. https://doi.org/10.3390/app14041565

Academic Editor: João M. F. Rodrigues

Received: 17 January 2024
Revised: 11 February 2024
Accepted: 12 February 2024
Published: 15 February 2024

1. Introduction

After the enterprise business is fully informationized, each terminal stores a large amount of text data, making it difficult for administrators to determine which text contains sensitive information, and it is impossible to apply control measures to all the text containing sensitive information. Some documents that do not have obvious confidential marks can easily evade regulatory measures and circulate freely inside and outside the company, ultimately leading to the leak of secrets. At present, most of the information security solutions still use firewalls [1], intrusion detection [2], network antivirus approaches [3], and other methods, which are relatively general and rough, lack the judgment and detection of information content, and cannot refine the management and control means. Data leakage prevention (DLP) [4] is the use of technical measures to prevent data assets from leaking out of an enterprise in violation of security policies. The classification and control of all documents in the terminal and the setting of protection measures can be a good way to block this leak vulnerability. The current data leakage protection methods mainly include internal data encryption [5], identity authentication [6], and data flow control [7]. The information leak prevention solution based on text classification [8] scans and identifies

all outgoing texts sent by terminals in real time; identifies the category of the text in time; checks whether the text contains sensitive keywords [9] and the secret level of the text, which can be used to detect and deal with the leak phenomenon in real time; and then deals with the text through relevant preset means, such as a warning, alarm, or blocking. Therefore, this results in higher performance requirements for real-time massive text classification algorithms.

Text classification means sorting a target text into a given category, and is widely used in sentiment analysis (SA) [10], news classification (NC) [11], natural language inference (NLI) [12], and other fields. It is a basic step to construct a text classification model by learning the categorical feature information of classified text data and then automatically classifying unclassified text. There are already many types of text classification methods, such as the decision tree (DT) algorithm [13], the naive Bayes (NB) algorithm [14], the support vector machine (SVM) algorithm [15], the k-nearest neighbor (kNN) algorithm [16], and classification algorithms based on deep learning [17]. The DT algorithm establishes a mapping between object attributes and object values, and classifies unlabeled texts by constructing decision trees. The basic idea of NB classification is to estimate the category probability of a given text using the joint probability of the phrase and category. SVM is a popular supervised learning algorithm. Goudjil M et al. selected a group of informative samples using the posterior probabilities provided by multi-class SVM classifiers to enhance the classification accuracy [18]. J Ababneh et al. investigated the performance of the three classification algorithms—kNN, DT, and NB—as classifiers on Saudi Press Agency datasets [19]. Classification models based on deep learning mainly include the convolution neural network (CNN) [20], the recurrent neural network (RNN) [21], and a combination of other related models. Shen, C. W. et al. used a direct citation network with cluster analysis in a hybrid bibliometric approach to depict the historiographic development of the technology-enhanced learning (TEL) research domain in higher education [22]. A comparison of the above typical algorithms is shown in Table 1.

Table 1. Comparison of the above typical algorithms.

Algorithm	Method Used	Datasets	Advantages	Disadvantages
DT [13]	Explore emotion classification with DT	Composed of 2924 news articles of April 2007 and February 2008 from Sina [www.sina.com.cn/society]	Easy to understand, suitable for small datasets	It does not perform well when dealing with data with strong correlation features
NB [14]	Explore NB automatic classification system	Chinese webpage	Low complexity and high efficiency	The dependency between features will affect the classification accuracy
SVM [15]	Implement SVM in classifying English documents	Literature from UCI library	Suitable for processing large datasets	Multiple parameters need to be tuned, and different parameter combinations lead to different classification results
kNN [16]	Combine kNN with an explicit semantic analysis approach	Collection of over 4 million documents comprising only titles and abstracts of articles extracted from the MEDLINE database	Easy to understand, suitable for massive classification scene	The computation is large, especially when the number of features is very large
CNN [20]	Offer a kind of short text classification model by CNN	(1) English movie review datasets (2) Chinese categorized dataset ChnSentiCorp	Suitable for processing high dimensional data	Need to adjust parameters, and need a large sample size
RNN [21]	Modify RNN with long short-term memory (LSTM) architecture	13 real-world datasets for text classification from CrowdFlower and DL4J	It can capture long-term dependencies in the sequence	The computational complexity is higher in the training and inference

Among these, the typical statistical classification model is the kNN model, which has a relatively simple classification process and is currently the most widely used. In the field of document confidentiality management, the kNN algorithm is insensitive to the existence of outliers in the classification process, so it does not need repeated training and learning, but only needs to refresh neighbor samples according to the update of confidential content; thus, continuous training of the model due to the update of confidential content is avoided. However, the kNN method has some shortcomings, such as the huge dimension of the feature vector, leading to a huge amount of computation, which affects the classification accuracy and classification efficiency [23]. Therefore, it is meaningful to improve and optimize the performance of current kNN classification.

As a spatial parallel processing technology [24], the AVX instruction set belonging to Single-Instruction-Multiple-Data (SIMD) instructions [25] can process large-scale Euclidean distance operations [26] and comparison and sorting operations, which makes it possible to use the Central Processing Unit (CPU) as an effective computing power for massive text classification. Therefore, based on the AVX-256 instruction set, which has high efficiency and good maturity [27], optimizing the weight calculation of the text feature vector, the Euclidean distance calculation, and the distance sorting between samples can be used to fully utilize the CPU's computing role in large-scale text classification processing.

Aiming at realizing real-time text classification, we calculate the weight of the feature vector based on the improved feature selection algorithm *tf-DE* [28], and identify the category of the text based on the kNN optimized classification algorithm. The rest of this paper is organized as follows: the feature vector selection algorithm and kNN algorithm, which are used to implement text preprocessing, text representation, feature selection, and text classification, are introduced in Section 2. To improve the weight calculation performance of the feature vector and the Euclidean distance calculation between samples, in Section 3, the AVX-256 instruction set is used to improve the construction of the feature vector, the vector weight calculation, and the Euclidean distance calculation. In Section 4, the accuracy of the text classification algorithm with the feature vector selection algorithm and kNN algorithm optimized by AVX-256 is tested via experiments. In Section 5, we discuss the conclusions of the experiment and make assumptions about the future work.

2. Materials and Methods

The model of text classification includes a deep learning model and a shallow learning model. The structure of the deep learning model is relatively complex, and it can directly learn and model text content without relying on manually obtained text features. RNNs are widely used in the field of natural language processing. Socher et al. at Stanford University introduced an RNN model that learns compositional vector representations for phrases and sentences of arbitrary syntactic type and length [29]. In the process of back propagation, the parameter updating of the RNN depends on the gradient. Long short-term memory (LSTM) is an improved model of the RNN that effectively alleviates the problem of gradient disappearance. Liu Z. et al. proposed an attention-based LSTM model for scene text detection, which can well handle scene text objects with arbitrary shapes [30].

However, the deep learning model is highly dependent on data and has the problem of weak domain adaptability. The structure of the shallow learning model is relatively simple and relies on manually obtained text features. Although the model parameters are relatively few, it can often show better results in complex tasks and has good domain adaptability. The kNN, as a classic machine learning-based text classification algorithm, is a shallow learning model. Its core idea is to extract the feature of the text into feature words to represent the text, and use the feature words to form a feature vector. The feature vectors are used to train the classifier, which can automatically classify the unclassified text. There are four parts in text classification, namely, preprocessing of text, representation of text, feature selection, and the classification algorithm.

2.1. Text Preprocessing

The purpose of text preprocessing is to transform text into a data structure that can be processed by computer, that is, to divide text into independent feature words. This paper adopts a dictionary-based string-matching method to preprocess text, which is based on the establishment of a unified dictionary table. Before starting to classify text, a keyword dictionary is defined. Based on the text content, text type, text level, and other information, the representative words in the text are extracted, these words are gathered together, and the category and level of each keyword are configured, so as to form a keyword dictionary. The dictionary diagram is shown in Figure 1.

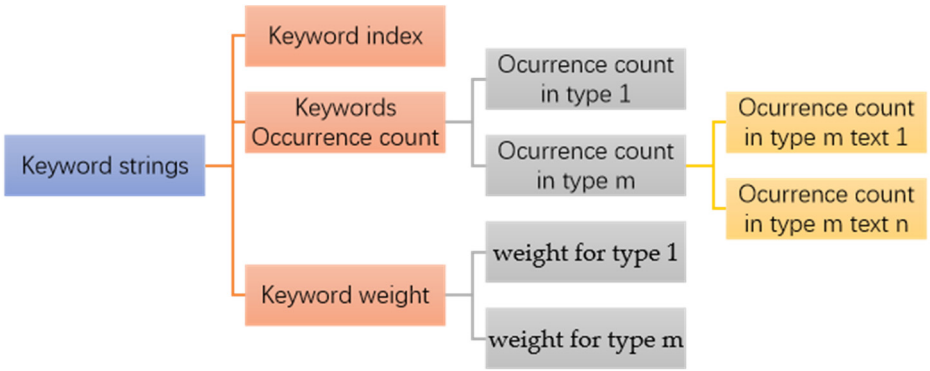

Figure 1. Diagram of the dictionary.

We construct a keyword dictionary according to high-frequency words in all categories of text. The construction method of the keyword dictionary is as follows.

(1) Go through all the documents in the training set, working on one text at a time.
(2) Firstly, the text is divided into words. Here, jieba, which is an important third-party Chinese word segmentation library in Python, is adopted as the word segmentation tool [31]. After word segmentation, further remove punctuation, single words, and stops.
(3) Then, the occurrence frequency of each word is counted and recorded.
(4) Finally, find the N words with the highest frequency as the high-frequency words by sorting in all text, and obtain the keyword dictionary.

Based on the keyword dictionary, text is checked by string matching. When a text is preprocessed, the text is first divided into several parts, and each part is matched with a dictionary entry according to a certain strategy, so as to match all keywords in the text. If the keyword is in the dictionary, the segmentation is successful; otherwise, continue splitting until the match is successful. The text scanning sequence of this method includes forward, reverse, and bidirectional, and the matching methods include maximum or minimum matching, word-by-word matching, and best matching [32]. In this paper, the data stream and keywords are compared in the binary layer. In order to support the text of various encoding modes, namely, ANSI, UTF-8, and Unicode [33], the three most commonly used encoding modes are used for keyword searching. Because the searching process traverses the entire file, the KMP searching algorithm [34] is used to ensure efficient searching. After keyword searching for each text, the results are saved in XML format, and the subsequent feature value selection and text classification algorithms use the keyword searching results in XML format as input.

Through the keyword occurrence times recorded in the dictionary, the best keyword can be selected according to the scale of the feature value in the subsequent feature value selection, and the required data can be provided for the feature weight calculation. After the above keyword scanning, each keyword appearing in the text and its occurrence time can be obtained. Text grading is based on the data value, sensitivity, influence, and distribution scope of keywords in the text. Different levels of data are processed in different ways when

outgoing distribution is controlled. Suitable feature word screening can simultaneously improve the classification performance and accuracy. This is because redundant feature words are not only unimportant, they also cause interference and misdirection in the classification process. Therefore, removing them can make the classification results more accurate and concentrated according to each type of text.

2.2. Text Representation

The function of text representation is to transform unstructured information into structured information that the computer can understand, that is, to represent the text as a suitable data structure for subsequent classification steps. The vector space model (VSM) method [35] is currently the most used and effective text representation in text processing applications. The VSM method represents the text as an n-dimensional vector, giving each word a certain weight in the word vector.

$$(w(t_1), w(t_2), \ldots, w(t_n)) \tag{1}$$

where t stands for feature word, and $w(t_i)$ (I = 1,2,. . .,n) represents the ith feature word's weight, which is computed by the grade and frequency of the feature word.

The representation effect of kNN depends heavily on the design of the weight function. With a proper weight function, kNN can well summarize the features of the text. The advantage of kNN is that it represents the natural language in the form of a vector, which is very conducive to the single instruction and multiple data processing of the vector. At the same time, because the text contains too many words, the dimension of the kNN vector is very large, and the dimension must be reduced through feature word selection.

2.3. Feature Selection

After text preprocessing and representation, the obtained feature words are still large in number, even after preprocessing and filtering, which leads to the high dimension of text representation vector. Therefore, it is important to use the feature selection algorithm to filter the feature words and thus reduce the dimension. Feature selection can not only reduce the computational complexity, but also help to improve the classification performance. In this paper, according to the limitation of the scale of the feature vector, the appropriate keyword range is selected, and the feature vector of the text is extracted and constructed according to the results of the keyword scanning. The construction of the feature vector is shown in Table 2.

Table 2. Construction of the feature vector.

Keyword 1 Feature		**Keyword 2 Feature**		**Keyword x Feature**	
weight for type 1	weight for type m	weight for type 1	weight for type m	weight for type 1	weight for type m

The classic feature selection algorithm *tf-idf* adopts lexical statistical features as the feature set [36], and term frequency (TF) and inverse document frequency (IDF) are used to represent the words' importance in the text. The weight expression of feature words in the *tf-idf* algorithm is shown below.

$$w_{ij} = tf_{ij} \cdot idf = tf_{ij} \cdot \log\left(\frac{N}{n} + 0.01\right) \tag{2}$$

where w_{ij} represents the weight of keyword t_j on classification C_i and describes the closeness between a keyword and this category; tf_{ij} represents "word frequency", that is, the sum of occurrence times in all documents in classification C_i of keyword t_j; *idf* represents "anti-document frequency", that is, the ratio between the number of documents and the number of documents containing keyword t_j. N represents the number of documents, n represents

the total number of documents containing t_j, and + 0.01 is used to prevent the situation where the weight is 0.

However, the *tf-idf* algorithm does not consider the distribution difference of words among different categories, and will give larger *idf* values to common words that widely appear in each category and smaller *idf* values to uncommon words concentrated in one category. Too low *idf* values of feature words in a category will result in a weight that is too low. On the other hand, the *tf-idf* algorithm ignores the distribution of words within the class. For example, some "different" keywords may appear in a category due to accidental factors, and within the document, the "different" keyword may appear frequently. In this case, the keyword should not represent the category. On the basis of analyzing some shortcomings of *tf-idf* algorithm, algorithm *tf-DE* is introduced. New statistics, Inter-Category Dispersion and Intra-Category Information Entropy, are introduced for feature selection [28].

A parameter is selected to describe the dispersion degree of the keyword distribution across multiple categories. Keywords with a more concentrated distribution in a certain category or a few categories are given higher values because such words obviously have better classification ability, while keywords evenly dispersed in each category without much classification representation ability should have relatively low values. Based on the statistical standard deviation formula, parameter D is defined as shown in the following.

$$D(t) = \begin{cases} \frac{\sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(d_i(t)-\overline{d(t)}\right)^2}}{\overline{d(t)}}, & \overline{d(t)} \neq 0 \\ 0, & \overline{d(t)} = 0 \end{cases} \tag{3}$$

where $d_i(t)$ represents the number of documents containing keyword t in category C_i, $\overline{d(t)}$ represents the average value of $d_i(t)$, and m represents the number of categories.

The information entropy E [37] in information theory is used to describe the distribution of keywords in a category. In any category, keywords that appear in many documents can better represent this category, and such terms should be selected to participate in the classification as far as possible. That is, the more evenly distributed keywords among the documents within the class should be given greater weight.

$$E(t, C_i) = -\sum_{j=1}^{n} e_j \tag{4}$$

where n represents the number of documents in C_i; e_j calculates the jth document in C_i, and its definition is as follows.

$$e_j = \begin{cases} \frac{Nd_j}{NC_i} \log_2 \frac{Nd_j}{NC_i}, & NC_i \neq 0 \text{ and } Nd_j \neq 0 \\ 0, & NC_i = 0 \text{ or } Nd_j = 0 \end{cases} \tag{5}$$

where Nd_i represents the number of occurrence times in the jth document of the category C_i of keyword t, and NC_i represents the number of occurrence times in all the documents of the category C_i of keyword t. The product of Inter-Category Dispersion and Intra-Category Information Entropy is used to replace the *idf* item in *tf-idf*, thus introducing the statistics of the distribution of feature words between and within categories, so that they can affect the final weight value; that is, the feature selection algorithm *tf-DE* is shown in the following.

$$w_{ij} = tf_{ij} \cdot D(t) \cdot E(t, C_i) \tag{6}$$

In this paper, parameters D and E are calculated respectively according to the keyword occurrence times in the dictionary. Similar keywords are selected to be clustered into a class, and this class is then used as the dimension of the vector space to represent the text. In this way, the problems of synonyms in the dictionary can be dealt with, so as to further reduce the dimension of the feature vector and reduce the complexity of the feature weight

calculation and distance calculation. The algorithm for feature vector extraction is shown in Algorithm 1, which is explained in detail below.

Algorithm 1: feature vector extraction algorithm for text

```
Input: trainText, keywordList
Output: keywordList
1. Function Load_trainText(trainText, keywordList)
2.   fileID, label = trainText.id, trainText.label
3.   keywordResult = KeywordsCheck(trainText, keywordList)
4.   for i = 1 to keywordResult.length
5.       id = GetKeyword(keywordResult[i])
6.        keywordList[id].type[label].file[fileID].count = keywordResult[i].count
7.        keywordList[id].type[label].count += keywordResult[i].count
Input: keywordList
Output: keywordList
8. Function Calc_weight(keywordList)
9.    for i = 1 to keywordList.length
10.       for j = 1 to M
11.          tfij = NC = keywordList[i].type[j]. count
12.          for k = 1 to keywordList[i].type[j].fileNum
13.               if (keywordList[i].type[j].file[k].count > 0) d[j] = tfij
14.             Nd = keywordList[i].type[j].file[k].count
15.             E[j] += Nd/NC * log(Nd/NC)
16.       D=Standard_deviation(d)
17.       for j = 1 to M
18.          wij = tfij * D * E[j]
Input: testText, keywordList
Output: featureVector
19. Function Get_featureVector (testText, keywordList)
20.    featureVector = InitFeatureVector(keywordList)
21.    keywordResult = KeywordsCheck(text, keywordList)
22.    for i = 1 to keywordResult.length
23.       id, weightVector = GetKeyword(keywordResult[i])
24.       featureVector[id] = weightVector
```

1. Reading training text set.
2. Obtain the training text ID and the category label.
3. Find keywords in the training text.
4. Iterate through the keywords in the training text.
5. Obtain keyword ID.
6. Count the occurrence number of keyword in this text.
7. Count the occurrence number of keyword in the category.
8. Feature vector weight calculation function.
9. Go through the keyword *i* in each dictionary.
10. Iterate through each category *j*.
11. Obtain the occurrence number of keyword *i* in category *j*, that means tf_{ij} and *NC*.
12. Go through each document *k* in the category
13. Count the number d_j of text containing keyword *i* in category *j*.
14. Obtain the number of occurrences *Nd* of keyword *i* in text *k* of category *j*.
15. Calculate the parameter *E* of keyword *i* for category *j*.
16. Calculate parameter *D* of keyword *i*.
17. Go through each category *j*.
18. Calculate the weight of keyword *i* to class *j*.
19. Construct a feature vector function of the target text.
20. The text vector space is initialized according to the size of the keywords.
21. Find the keywords for the target text based on the keyword list.

22. Go through keywords in the target text.
23. Obtain the index and weight vector of the *i*th keyword.
24. Construct feature vector for text.

Among the many feature selection algorithms, *tf-OR* (term frequency and odds ratio) and *tf-rf* (term frequency and relevance frequency) are excellent algorithms. The weight representation of *tf-OR* is shown in Equation (7) and the weight representation of *tf-rf* is shown in Equation (8), where we use C_i to represent the category in which keyword t occurs currently, $\overline{C_i}$ to represent all other categories, tp to represent the number of documents that contain keyword t in category C_i, fp to represent the number of documents that do not contain keyword t in category C_i, fn to represent the number of documents that contain keyword t in category $\overline{C_i}$, and tn to represent the number of documents that do not contain keyword t in category $\overline{C_i}$ [28]. In Section 4.4, feature selection algorithms *tf-DE*, *tf-idf*, *tf-OR*, and *tf-rf* are selected for comparative testing.

$$w_{ij} = tf_{ij} \cdot \log \frac{tp \cdot tn}{fp \cdot fn} \tag{7}$$

$$w_{ij} = tf_{ij} \cdot \log\left(2 + \frac{tp}{fn}\right) \tag{8}$$

2.4. The kNN Classification Algorithm

The guiding idea of kNN is that if most of the k sample points nearest to the sample point belong to this certain category, then the sample point should also belong to that category, which means kNN is used in the selection of the distance function. The commonly used distance functions include Euclidean distance, Chebyshev distance, and Canberra distance [38]. In this paper, Euclidean distance is selected as the distance function to calculate the similarity between samples, which is shown in the following.

$$d(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2} \tag{9}$$

3. Text Classification Algorithm Optimization Based on AVX-256

3.1. Proposed Text Classification Process

Given an unlabeled text d, the classification system searches for k adjacent labeled documents that are closest to it in the training set, and then determines the category of text d according to the classification labels of the adjacent texts. The kNN classification implementation process is as follows.

(1) Transform the training set into a vector space model representation and calculate the weight of each feature.
(2) Convert the unlabeled text d in a similar way as in step (1) and calculate the weight of the corresponding phrase element.
(3) Calculate the distance between text d and each text in the training set.
(4) Identify the k training documents with the smallest distance from text d.
(5) According to the category attributes of the first k training documents, text d is generally classified as the most frequent sample category in the k training documents.

In essence, the kNN classification model has no specific training and learning process, and its classification process only calculates the similarity between the unlabeled text and each training set sample. Therefore, the time and space complexity of the kNN algorithm is high. With the increase in the number of training samples, the storage resource consumption of classification is large and the time cost is high. In order to improve the classification speed, we use the AVX instruction set to realize the simultaneous calculation of multiple feature value distances and reduce the instruction consumption of kNN, so as to improve the performance of the kNN algorithm and meet the requirements of fast classification of

massive texts. Figure 2 shows the procedure of the text classification algorithm accelerated with the use of AVX-256, which is elaborated in Algorithm 2 with a step-by-step explanation of the procedure.

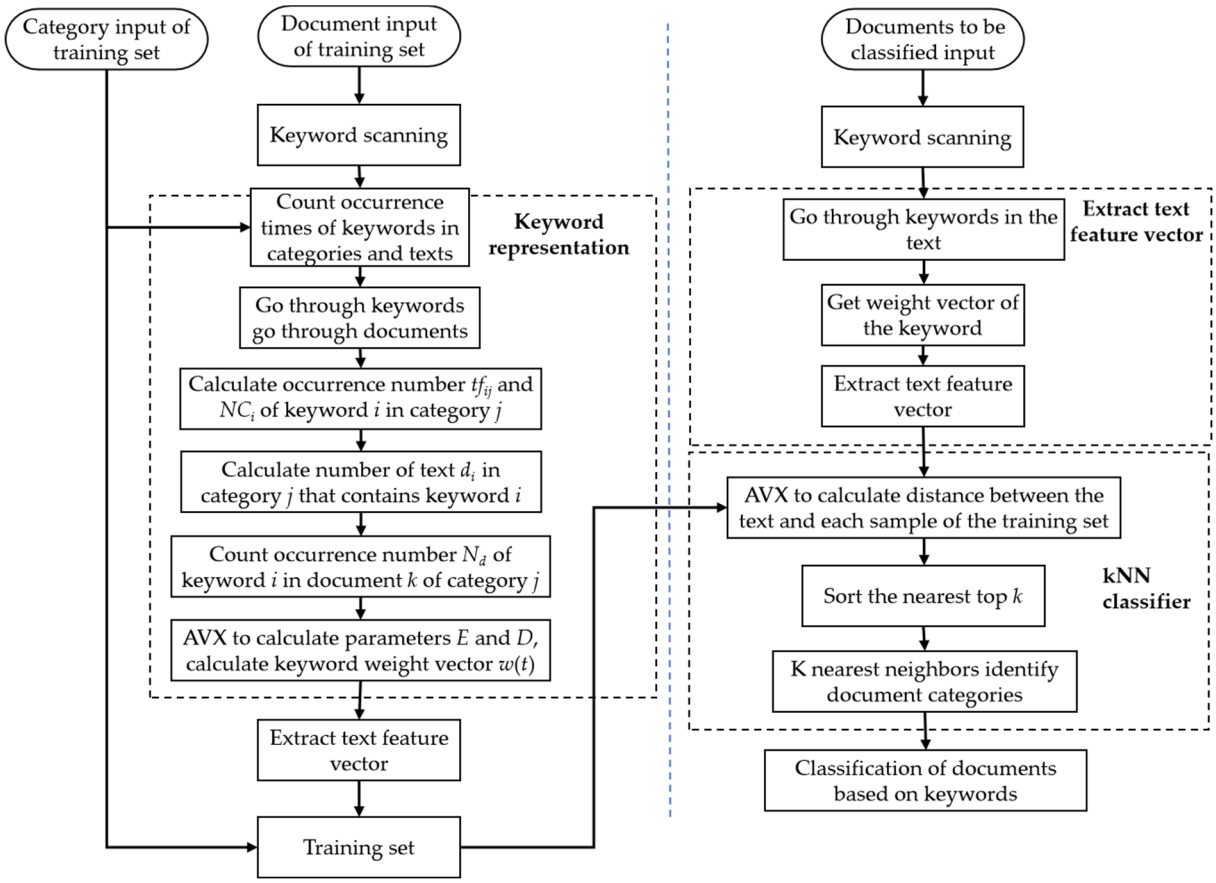

Figure 2. The procedure of text classification algorithm with AVX-256 acceleration.

Algorithm 2: text classification based on AVX-256

```
Input: trainSet, testText, keywordList
Output: type
1. keywordList = LoadKeyword()
2. for i = 1 to trainSet.length
3.     Load_trainText (trainSet.text[i], keywordList)
4. Calc_weight(keywordList)
5. for i = 1 to trainSet.length
6.      trainSet.vector[i] = Get_featureVector (trainSet.text[i], keywordList)
7. featureVector = Get_featureVector(testText, keywordList)
8. for i = 1 to trainSet.length
9.      neighbor[i].distance = CalculateDistance(featureVector, trainSet.vector[i])
10.      neighbor[i].type = trainSet.type[i]
11. k_nearest_neighbor = sort_k(neighbor)
12. type = k_nearest_neighbor.most_type
```

1. Read keyword list.
2. Go through the training set.
3. Extract the key words of the *i*th training text.
4. Feature selection, calculate vector weights.
5. Go through the training set.
6. Extract the feature vector of the training text and construct the training sample set.

7. Extract the feature vector of the text to be classified
8. Go through the training set.
9. Calculate the distance between the text to be classified and the *i*th training text.
10. Obtain the classification type of the *i*th training text.
11. Sort by neighbor distance.
12. Identify the target document category based on the category of the k-nearest neighbors.

3.2. Text Classification Algorithm Optimization Based on AVX-256

Computer instructions can be divided into scalar instructions and vector instructions [39], where scalar instructions can only operate on one data point at a time. As the core technology of vector instruction, SIMD can use one instruction to process multiple data points in parallel. Compared with traditional single data instructions, SIMD has fine parallelism and can increase the computation speed exponentially in data-intensive matrix vector operations. For traditional kNN classification calculations, scalar instruction is still used in floating-point calculations, which wastes vector computation resources in the CPU. SIMD technology can be used to achieve efficient utilization of CPU computing resources. AVX-256 can process eight 32-bit data at a time, increasing the Euclidean distance calculation speed by nearly 4–8 times. Assuming that the dimensionality of the document feature vector is n and the number of samples in the training set is p, the arithmetic complexity of kNN is O(n*p), while the arithmetic complexity after optimization by AVX is O(n*p/4). Based on AVX-256, vector construction and operation can be realized; thus, the kNN text classification algorithm is optimized. This method can realize the simultaneous calculation of multiple feature value distances and reduce the instruction operation consumption of kNN, so as to improve the efficiency of the kNN algorithm and meet the fast classification requirements of large-scale texts.

In the dictionary, the number of keyword occurrences in each text can be represented by a 32-bit wide number, and an AVX instruction can simultaneously achieve eight feature parameter operations for AVX-256, supporting 256-bit wide vector operations.

(1) Occurrence vector construction: Use the instruction _mm256_loadu_ps to construct a vector, which includes the occurrence of a keyword in eight documents.
(2) Calculation of characteristic parameter e_j: Through _mm256_div_ps and _mm256_log2_ps instructions to achieve vector division and log calculation, the calculation of eight values is completed in each operation.
(3) Calculation of characteristic parameter E: By _mm256_add_ps instruction, the addition of vectors is realized and the information entropy E within the category is calculated.

The results of text classification feature value and distance are represented by 64-bit wide numbers, using one AVX instruction to realize the operation of four feature value distances at the same time. The kNN algorithm with AVX-256 acceleration is implemented as follows.

(1) Initialize the distance vector vec_sum and temporary vector vec_d, using _mm256_setzero_pd, which includes four 64-bit data elements to record the distances of four feature values.
(2) Construct the feature vector using instruction _mm256_loadu_pd. Use the instruction _mm256_loadu_pd to copy four feature values to construct the feature vector vec_a for training text and vec_b for text to be classified. The four elements of vector vec_a are represented by a_1, a_2, a_3 and a_4, at the same time, the four elements of vector vec_b are represented by b_1, b_2, b_3 and b_4. Each vector's composition structure is shown in Figure 3.

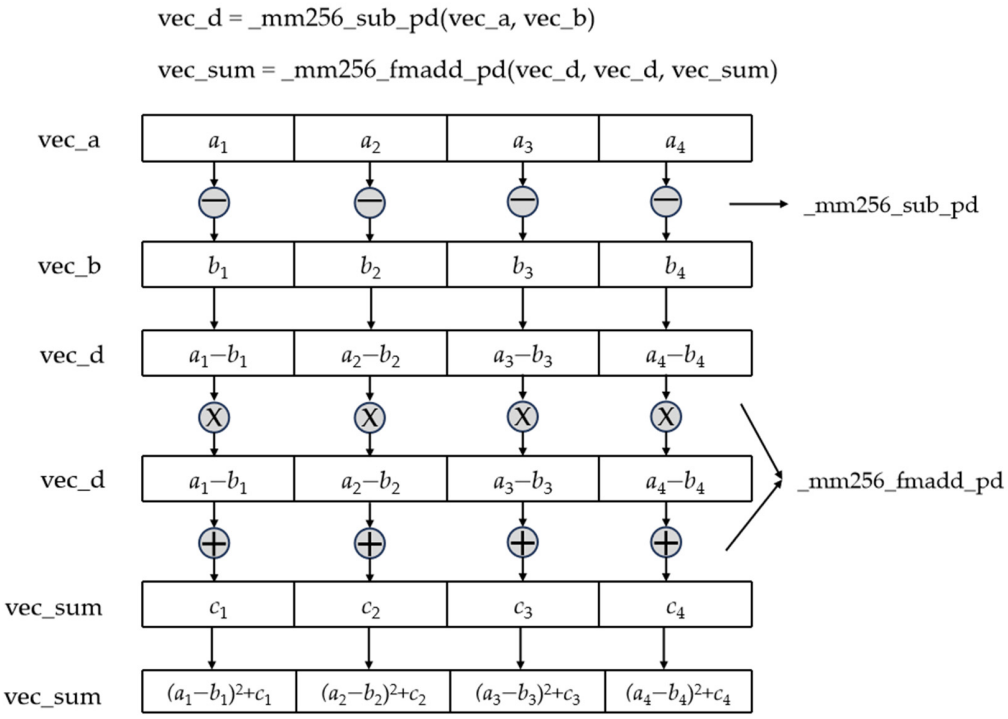

Figure 3. The composition structure of each vector.

(3) Process the sum of squares using an AVX instruction to complete multiple feature value operations, which means using the instruction _mm256_sub_pd to achieve vector subtraction, and using instruction _mm256_fmadd_pd to achieve multiplication of the difference vector and addition of the distance vector. Each operation completes the calculation of four feature value distances, thus improving the performance of kNN.

$$\begin{array}{c} \text{diff\_v = \_mm256\_sub\_pd(a\_v, b\_v)} \\ \text{sum\_v = \_mm256\_fmadd\_pd(diff\_v, diff\_v, sum\_v)} \end{array} \quad (10)$$

(4) Count distance, and sum the four distance results in the distance vector to obtain the sum of squares of all feature differences, and then obtain the Euclidean distance between the text to be classified and the samples.

4. Analysis of Experiments and Results

4.1. Experimental Setup

For the text classification algorithm described in Section 3, experiments using Windows 11 Professional Edition operating system, 11th Gen Intel® Core™ i9-11950 H @ 2.60 GHz processor, Visual Studio 2022 development platform, and C++ programming language were set up to inspect the accuracy and efficiency of the text classification algorithm with AVX-256 acceleration. Two implementation methods of the text classification were designed. In the first method, the feature selection algorithm and kNN classification algorithm were realized with a traditional Single-Instruction-Single-Data (SISD) stream, whereas in the comparison method, SIMD instructions were used to complete the operations of multiple feature parameters and multiple feature value distances at once. Under the same training set, the two versions of the algorithm have the same classification results for the same target text.

Since our research is mainly expected to be applied to the classification of Chinese texts, we choose THUCNews as the experimental dataset [40], which contains 740,000 news documents from 2005 to 2011, by filtering the historical data of the Sina news RSS subscription channel. Five categories were selected to classify text and inspect the comparison method in terms of improving the performance and efficiency of text classification, including finance, education, sports, technology, and entertainment.

4.2. Criteria for Evaluating Statistical Results

The evaluation algorithm F1-measure is adopted to evaluate the statistical results [41]. Firstly, four parameters representing the number of documents of a class are defined, and the specific meaning is as follows.

A: The kNN classification algorithm marks these texts as a category, and they belong to that category exactly. *B*: The kNN classification algorithm marks these texts as not belonging to a category, whereas, in fact, they should belong to that category. *C*: The kNN classification algorithm marks these texts as a category that they do not actually belong to. *D*: The kNN classification algorithm marks these texts as not belonging to a category, and they do not belong to that category exactly.

For any one category, the classifier accuracy *P* and recall *R* are defined as shown below.

$$P = \frac{A}{A + C}, \; R = \frac{A}{A + B} \tag{11}$$

A new parameter *F1-measure* is defined to integrate the meaning of the two parameters *P* and *R*, as shown below.

$$F1 - measure = \frac{2RP}{R + P} \tag{12}$$

4.3. Experimental Procedure

In this section, the effectiveness of the text classification algorithm under different feature vector scales is verified, and the performance gap between the traditional SISD-based method and the AVX-256 acceleration method is compared.

(1) Construct the dataset. Five categories were selected from the total corpus of the experiment, and 2000 articles in each category were selected as the training set and 1000 articles as the test set. The composition of the dataset is shown in Table 3.

Table 3. Composition of dataset.

Category	Finance	Education	Technology	Sports	Entertainment	Total
training set	2000	2000	2000	2000	2000	10,000
test set	1000	1000	1000	1000	1000	5000

(2) Text preprocessing and presentation. According to 10,000 training documents, the selected high-frequency words were used to construct a keyword dictionary of different sizes.
(3) Feature selection. According to 10,000 training documents, the occurrence times of each keyword in each text were counted, and the weight of the feature vector was calculated according to the *tf-DE* algorithm.
(4) Training sample set. In this step, text representation and preprocessing were carried out on the training sample set, that is, the previous steps (2) and (3) were carried out on the training sample set to obtain the vector set of the training samples.
(5) Classification test. First of all, text representation and preprocessing of the test set were required; this means that the previous steps (2) and (3) of the test set were executed to obtain the vector set of the test set.
(6) Calculate the distance between the text to be tested and each sample in the training sample set, so as to find the nearest *k* neighbors, and then obtain the category type of the test text according to the types of *k* neighbors.
(7) Obtain results of statistical classification.

4.4. Results Analysis

The traditional SISD-based method and the AVX-256 acceleration method of the text classification algorithm are applied as contrasting experiments to observe and analyze the

performance of feature value selection, distance calculation, and nearest neighbor sorting of the test dataset.

When the feature selection algorithm *tf-DE* is selected for testing, the multi-classification effects of the kNN and SVM are shown in Table 4. The comparison results show that the *F1-measures* of kNN and SVM models are comparable.

Table 4. Multi-classification effects of kNN and SVM.

Type \ Indicator	Accuracy *P*		Recall *R*		*F1-Measure*	
	kNN	**SVM**	**kNN**	**SVM**	**kNN**	**SVM**
finance	95.7%	91.1%	88.3%	98.0%	91.8%	94.4%
education	87.6%	95.3%	85.7%	73.5%	86.6%	83.0%
technology	82.7%	94.8%	90.8%	85.0%	86.6%	89.6%
sports	96.1%	83.9%	95.8%	99.5%	96.0%	91.0%
entertainment	95.2%	94.7%	97.1%	99.5%	96.2%	97.0%
average	91.5%	92.0%	91.5%	91.1%	91.4%	91.0%

The test results are shown in Figure 4, which shows that as the lexical diversity of the text in the entertainment category is low, and that high-frequency words in the entertainment category, such as film, director, and box office, can better represent the entertainment category. Therefore, when the feature vector dimension is small, a higher *F1-measure* value can be achieved, and the classification accuracy is as high as 95.5% shown by the red line. However, in the text of the education category, the lexical diversity is high, and when the feature vector dimension is small, the accuracy rate is only 70.9% shown by the gray line. However, when the feature vector dimension is larger, the classification accuracy of the education category text can also achieve a higher *F1-measure*, of 85.9%.

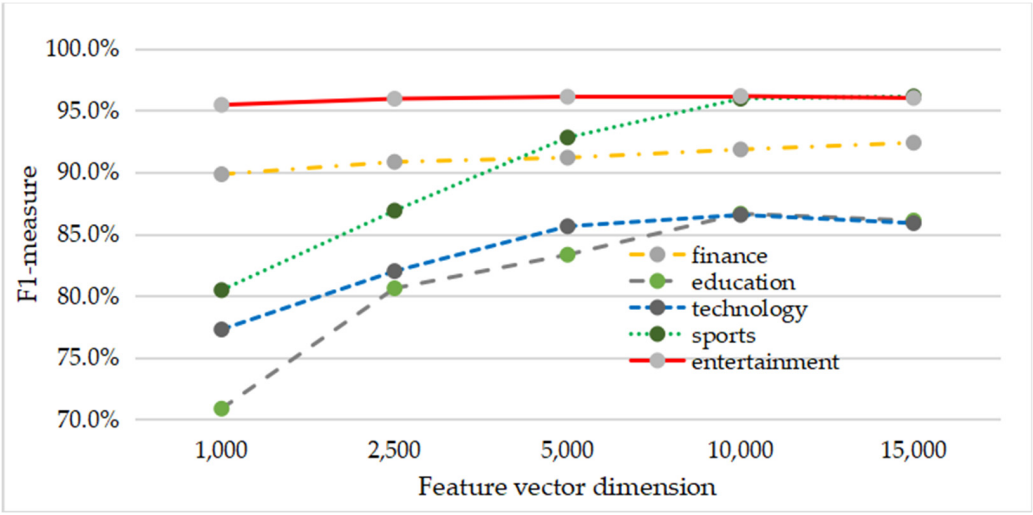

Figure 4. Test results of text classification algorithm under *tf-DE*.

When feature selection algorithms *tf-DE*, *tf-idf*, *tf-OR*, and *tf-rf* are used for comparative testing, the test results show that as the number of feature words increases, the classification effect of the text classification algorithm will increase, as shown in Figure 5. In the *tf-DE* algorithm, when the feature vector dimension is increased from 1000 to 15,000, *F1-measure* also increases from 86.3% to 93.5%. When the feature vector dimension is small, the *tf-DE* algorithm can hardly reflect the advantages. We believe that this is due to the diversity of the Chinese vocabulary; this makes it difficult to gather a certain number of keywords representing the features of documents, so Chinese text needs more feature words to judge its category. Therefore, our algorithm can only reflect its advantages when the number

of feature words is large. When the feature vector dimension exceeds 5000, the *tf-DE* algorithm begins to show certain advantages.

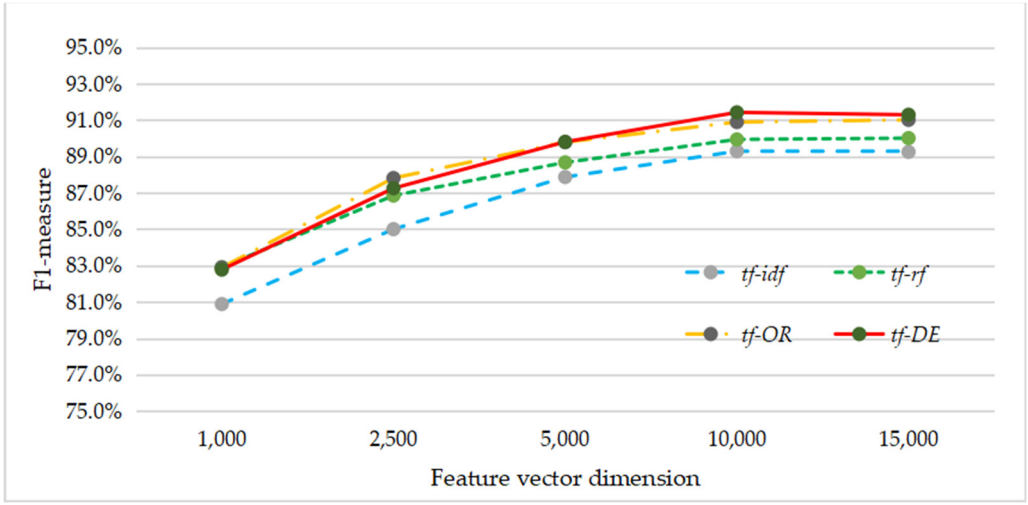

Figure 5. Simulation results of text classification algorithm under *tf-DE*, *tf-idf*, *tf-OR*, and *tf-rf*.

In terms of text classification performance, the classification time also increases linearly with the increase in feature vector dimension. Under the same vector dimension, the time consumed by the AVX-256 acceleration method is greatly reduced to about 55% of that of the traditional SISD-based method. For a test set containing 5000 documents, when the feature vector dimension is 15,000, the classification time of the kNN algorithm decreases from 748 s to 403 s. Table 5 illustrates a running time comparison of the kNN algorithm between the two methods under conditions of different feature vector dimensions. Figure 6 shows the bar chart of the running time comparison, which is an intuitive presentation of Table 5.

Table 5. Running time comparison.

Feature Vector Dimension	Processing Time (ms)		Optimize Ratio
	SISD-Based	AVX-256 Acceleration	
1000	52,459	27,259	52.0%
2500	122,964	67,877	55.2%
5000	243,153	139,973	57.6%
10,000	502,294	275,005	54.7%
15,000	748,506	403,069	53.8%

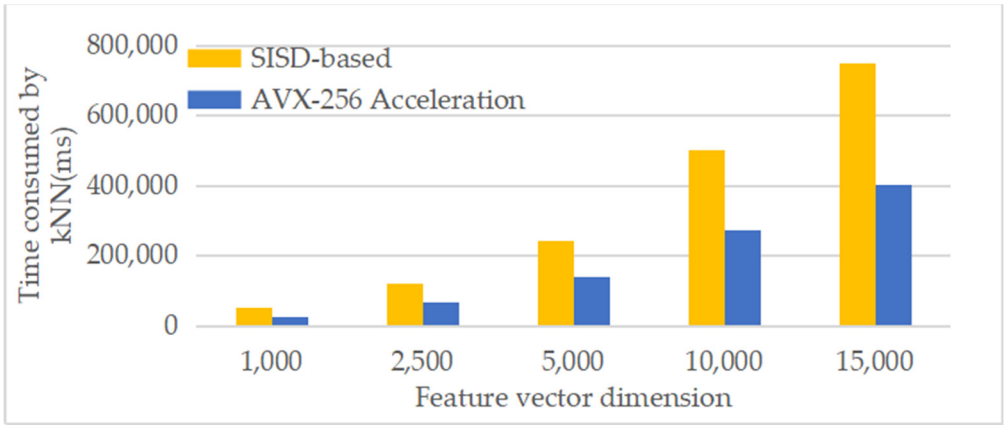

Figure 6. Bar chart of running time comparison.

5. Conclusions and Future Work

This paper describes a method for optimizing the automatic classification of sensitive electronic documents in enterprises and organizations to provide different levels of disclosure risk assessment by setting a unified inspection strategy. The proposed approach utilizes a combination of keyword retrieval and the kNN classification algorithm to standardize and enhance confidentiality management. In view of the circumstances, namely that the existing feature selection algorithm *tf-idf* ignores the distribution difference of words among and within different categories, and existing text classification algorithms based on kNN have high time and space complexity and large storage resource consumption, an optimization method is proposed to improve the performance of the feature selection algorithm and the kNN algorithm using AVX-256 to achieve real-time classification of massive texts. The features of the text are described and stored by vectors, and the operations that need to be performed separately by data are replaced by one vector operation. Firstly, the keyword dictionary and text feature vector are described in vectors, and the data vector is constructed, analyzed, and operated via SIMD instruction. Secondly, the weight operation and distance operation of multiple data are simplified into a vector operation to reduce the consumption of the CPU's computing resources. The algorithm with AVX-256 acceleration proposed in this paper improved the efficiency of high-dimensional feature vector computation performance, and the obtained classification results are comparable with those of the traditional classification method, SVM.

Although the kNN algorithm is simple and efficient, its accuracy needs to be improved. The deep learning models do not rely on artificially acquired text features, and can directly learn and model the text content, which is a great advance compared with the shallow learning models. For example, an RNN can recursively learn text semantic and syntactic tree structures without the need to set artificial features. A CNN can simultaneously use different convolution kernels for convolution operations on text sequences. There are also many studies that combine multiple deep learning models. In the follow-up work, the deep learning model for real-time text classification will be studied to improve multi-classification. Due to the limited computing power of CPUs and the limited memory capacity of Graphic Processing Units (GPUs), the scale of deep learning network models that they can implement is greatly limited. Based on the AVX instruction set, the hybrid operation strategy of the CPU and GPU in text classification algorithms will also be explored, so as to present the computing function and advantages of the CPU in neural network training. In addition, Chinese and English word segmentation methods are different; Chinese word segmentation needs to consider the granularity, which is more difficult. The jieba segmentation has poor performance in English classification. Because English has spaces as separators and English words have various forms, it is necessary to use segmentation tools applicable to English to classify English texts in the follow-up research.

Author Contributions: Conceptualization, software, validation, and writing—original draft L.T.; methodology, J.Y.; supervision, F.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Cheng, Y.; Wang, W.; Min, G.; Wang, J. A new approach to designing firewall based on multidimensional matrix. *Concurr. Comput. Pr. Exp.* **2013**, *27*, 3075–3088. [CrossRef]
2. Umer, M.F.; Sher, M.; Bi, Y. Flow-based intrusion detection: Techniques and challenges—ScienceDirect. *Comput. Secur.* **2017**, *70*, 238–254. [CrossRef]

3. Shukla, J.; Singh, G.; Shukla, P.; Tripathi, A. Modeling and analysis of the effects of antivirus software on an infected computer network. *Appl. Math. Comput.* **2014**, *227*, 11–18. [CrossRef]
4. Van der Kleij, R.; Wijn, R.; Hof, T. An application and empirical test of the Capability Opportunity Motivation-Behaviour model to data leakage prevention in financial organizations—ScienceDirect. *Comput. Secur.* **2020**, *97*, 101970. [CrossRef]
5. Jiang, P.; Ning, J.; Liang, K.; Dong, C.; Chen, J.; Cao, Z. Encryption Switching Service: Securely Switch Your Encrypted Data to Another Format. *IEEE Trans. Serv. Comput.* **2018**, *2018*, 1. [CrossRef]
6. Liu, H.; Ai, M.; Huang, R.; Qiu, R.; Li, Y. Identity authentication for edge devices based on zero-trust architecture. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e7198. [CrossRef]
7. Rong-na, X.; Hui, L.; Guo-zhen, S.; Yun-chuan, G.; Ben, N.; Mang, S. Provenance-based data flow control mechanism for Internet of things. *Trans. Emerging Tel. Tech.* **2021**, *32*, e3934. [CrossRef]
8. Deng, J.; Cheng, L.; Wang, Z. Attention-based BiLSTM fused CNN with gating mechanism model for Chinese long text classification. *Comput. Speech Lang.* **2021**, *68*, 101182. [CrossRef]
9. Gao, X.; Yu, J.; Chang, Y.; Wang, H.; Fan, J. Checking Only When It Is Necessary: Enabling Integrity Auditing Based on the Keyword with Sensitive Information Privacy for Encrypted Cloud Data. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 3774–3789. [CrossRef]
10. Chen, W.; Xu, Z.; Zheng, X.; Yu, Q.; Luo, Y. Research on Sentiment Classification of Online Travel Review Text. *Appl. Sci.* **2020**, *10*, 5275. [CrossRef]
11. Sun, N.; Du, C. News Text Classification Method and Simulation Based on the Hybrid Deep Learning Model. *Complexity* **2021**, *2021*, 1–11. [CrossRef]
12. Lan, A.G.J.; Paraboni, I. Text-and author-dependent moral foundations classification. *New Rev. Hypermedia Multimed.* **2022**, *28*, 18–38. [CrossRef]
13. Winster, S.G.; Kumar, M.N. Automatic classification of emotions in news articles through ensemble decision tree classification techniques. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 5709–5720. [CrossRef]
14. Gao, H.; Zeng, X.; Yao, C. Application of improved distributed Naive Bayesian algorithms in text classification. *J. Supercomput.* **2019**, *75*, 5831–5847. [CrossRef]
15. Luo, X. Efficient English text classification using selected machine learning techniques. *Alex. Eng. J.* **2021**, *60*, 3401–3409. [CrossRef]
16. Dramé, K.; Mougin, F.; Diallo, G. Large scale biomedical texts classification: A kNN and an ESA-based approaches. *J. Biomed. Semant.* **2016**, *7*, 1–12. [CrossRef]
17. Minaee, S.; Kalchbrenner, N.; Cambria, E.; Nikzad, N.; Chenaghlu, M.; Gao, J. Deep learning--based text classification: A comprehensive review. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–40. [CrossRef]
18. Goudjil, M.; Koudil, M.; Bedda, M.; Ghoggali, N. A novel active learning method using SVM for text classification. *Int. J. Autom. Comput.* **2018**, *15*, 290–298. [CrossRef]
19. Ababneh, J. Application of Naïve Bayes, decision tree, and K-nearest neighbors for automated text classification. *Mod. Appl. Sci.* **2019**, *13*, 31. [CrossRef]
20. Wang, H.; He, J.; Zhang, X.; Liu, S. A short text classification method based on N-gram and CNN. *Chin. J. Electron.* **2020**, *29*, 248–254. [CrossRef]
21. Du, J.; Vong, C.M.; Chen, C.L.P. Novel efficient RNN and LSTM-like architectures: Recurrent and gated broad learning systems and their applications for text classification. *IEEE Trans. Cybern.* **2020**, *51*, 1586–1597. [CrossRef]
22. Shen, C.W.; Ho, J.T. Technology-enhanced learning in higher education: A bibliometric analysis with latent semantic approach. *Comput. Hum. Behav.* **2020**, *104*, 106177. [CrossRef]
23. Zhang, S.; Li, X.; Zong, M.; Zhu, X.; Cheng, D. Learning k for kNN classification. *ACM Trans. Intell. Syst. Technol. (TIST)* **2017**, *8*, 1–19. [CrossRef]
24. Geng, T.; Waeijen, L.; Peemen, M.; Corporaal, H.; He, Y. MacSim: A MAC-Enabled High-Performance Low-Power SIMD Architecture. In Proceedings of the 2016 Euromicro Conference on Digital System Design (DSD), Limassol, Cyprus, 31 August–2 September 2016; pp. 160–167. [CrossRef]
25. Jakobs, T.; Kratzsch, S.; Rünger, G. Analyzing Data Reordering of a combined MPI and AVX execution of a Jacobi Method. In Proceedings of the 2023 31st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Naples, Italy, 1–3 March 2023; pp. 159–163. [CrossRef]
26. Salihu, S.A.; Onyekwere, I.P.; Mabayoje, M.A.; Mojeed, H.A. Performance Evaluation of Manhattan and Euclidean Distance Measures for Clustering Based Automatic Text Summarization. *FUOYE J. Eng. Technol.* **2019**, *4*(1), 135–139. [CrossRef]
27. Kim, T.; Hwang, C.; Park, K.S.; Lin, Z.; Cheng, P.; Miao, Y.; Ma, L.; Xiong, Y. Accelerating gnn training with locality-aware partial execution. In Proceedings of the 12th ACM SIGOPS Asia-Pacific Workshop on Systems, Hong Kong China, 24–25 August 2021; pp. 34–41.
28. Yi, J.; Yang, G.; Wan, J. Category Discrimination Based Feature Selection Algorithm in Chinese Text Classification. *J. Inf. Sci. Eng.* **2016**, *32*, 1145–1159.
29. Socher, R.; Huval, B.; Manning, C.D.; Ng, A.Y. Semantic compositionality through recursive matrix-vector spaces. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju Island, Republic of Korea, 12–14 July 2012; pp. 1201–1211.

30. Liu, Z.; Zhou, W.; Li, H. AB-LSTM: Attention-based bidirectional LSTM model for scene text detection. *ACM Trans. Multimed. Comput. Commun. Appl. (TOMM)* **2019**, *15*, 1–23. [CrossRef]
31. Liu, K.; Ergu, D.; Cai, Y.; Gong, B.; Sheng, J. A new approach to process the unknown words in financial public opinion. *Procedia Comput. Sci.* **2019**, *162*, 523–531. [CrossRef]
32. Wang, D.; Su, J.; Yu, H. Feature extraction and analysis of natural language processing for deep learning English language. *IEEE Access* **2020**, *8*, 46335–46345. [CrossRef]
33. Hilal, T.A.; Hilal, H.A. Arabic text lossless compression by characters encoding. *Procedia Comput. Sci.* **2019**, *155*, 618–623. [CrossRef]
34. Gurung, D.; Chakraborty, U.K.; Sharma, P. Intelligent predictive string search algorithm. *Procedia Comput. Sci.* **2016**, *79*, 161–169. [CrossRef]
35. Al-Anzi, F.S.; AbuZeina, D. Beyond vector space model for hierarchical Arabic text classification: A Markov chain approach. *Inf. Process. Manag.* **2018**, *54*, 105–115. [CrossRef]
36. Zheng, L.; Wang, H.; Gao, S. Sentimental feature selection for sentiment analysis of Chinese online reviews. *Int. J. Mach. Learn. Cybern.* **2018**, *9*, 75–84. [CrossRef]
37. Ali, A.; Naeem, S.; Anam, S.; Ahmed, M.M. Entropy in Information Theory from Many Perspectives and Various Mathematical Models. *J. Appl. Emerg. Sci.* **2022**, *12*, 156–165.
38. Eminagaoglu, M. A new similarity measure for vector space models in text classification and information retrieval. *J. Inf. Sci.* **2020**, *48*, 463–476. [CrossRef]
39. Chen, Z.; Kaeli, D. Balancing scalar and vector execution on gpu architectures. In Proceedings of the 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Chicago, IL, USA, 23–27 May 2016; pp. 973–982.
40. Shao, D.; Li, C.; Huang, C.; Xiang, Y.; Yu, Z. A news classification applied with new text representation based on the improved LDA. *Multimed. Tools Appl.* **2022**, *81*, 21521–21545. [CrossRef]
41. Rehman, A.; Javed, K.; Babri, H.A. Feature selection based on a normalized difference measure for text classification. *Inf. Process. Manag.* **2017**, *53*, 473–489. [CrossRef]