Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 228 (2023) 195–202

Procedia Computer Science

www.elsevier.com/locate/procedia

3rd International Conference on Machine Learning and Big Data Analytics for IoT Security and Privacy

Optimized Algorithm Design for Text similarity Detection Based on Artificial Intelligence and Natural Language Processing

Zhe Liu[a,] *, Jiajia Zhu[a], Xinyun Cheng[a], Qingqiang Lu[b]

[a]*State Grid Jiangsu Elecric Power Co., Ltd Information and Telecommunication Branch, Nanjing 210024, Jiangsu, China*
[b]*Swordfish Intelligent Technology (Nanjing) Co., Ltd, Nanjing 210000, Jiangsu, China*

Abstract

This paper presents a Text similarity detection method based on artificial intelligence and natural language processing. The method combines statistical machine learning and deep learning techniques and designs six models from three perspectives: character-level, word-level, and semantic-level. These models include the diff model based on machine learning, cosine similarity model, Jaccard model, TF-IDF model, as well as the SimCSE and SBERT models based on deep learning methods. To fully leverage the characteristics of these models, three scenarios are designed to calculate the similarity scores based on experience and multiple experimental results. The results show that calculating the similarity scores using these three scenarios not only achieves high accuracy but also requires fewer computational resources. As deep learning and natural language processing technologies continue to advance, Text similarity detection methods based on artificial intelligence and natural language processing will continue to be improved and play a more significant role in practice. Future research can explore more models and algorithms to enhance the accuracy and robustness of plagiarism detection.

Keywords: Text Detection; AI; NLP; Algorithms; Robustness

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000 .
E-mail address: liuz21@js.sgcc.com.cn

1877-0509
10.1016/j.procs.2023.11.023

1. Introduction

Statistical methods and machine learning models were pioneered by Bayes' theorem in 1763, and they have a long history. Under the motivation of facing challenges, they have continued to grow and develop. The first neural network machine was born in 1951. Since then, deep learning(DL) has experienced ups and downs [1][2]. In recent years, there has been a peak in deep learning models for two reasons: the enormous amount of data generated daily in the era of big data and the tremendous computational power obtained using GPUs and TPUs. Nowadays, in most tasks, deep learning models perform on par with human experts [3][4].

From 1986 to the 21st century, deep learning rapidly developed as an important part of machine learning, attracting increasing attention from researchers. Starting from computer vision tasks [5][6] and now in the interdisciplinary task of Natural Language Processing (NLP) [7][8], deep learning has demonstrated remarkable performance. Before the advent of deep learning, machine learning tasks typically relied on simple models such as Support Vector Machines (SVM) [9] or Logistic Regression [10]. This led to inevitable issues in data modeling, such as high-dimensional feature spaces. Additionally, when modeling NLP tasks with such models, additional syntactic features or manually crafted features had to be added, consuming significant human and material resources. Moreover, manually crafted features have limitations and make it challenging to achieve superior model performance. In recent years, the emergence of deep learning has addressed this problem. Researchers can now utilize deep neural networks to learn hierarchical features of constructed data, extract key information from words, sentences, or paragraphs in large-scale data, and aggregate this rich information to construct more powerful models.

Text matching [11] aims to uncover the semantic features embedded in texts to determine their contradiction or similarity. As one of the core tasks in the field of Natural Language Processing, text matching has broad applications in everyday life. With the continuous development of internet technology, the daily generation of data in the big data environment has skyrocketed. People need to find the information they need from this vast amount of data. Therefore, achieving efficient matching between two pieces of text regarding their relevance or contradiction is of great significance in addressing people's knowledge needs in daily life.

In the era of big data, traditional naive statistical models and machine learning algorithms are no longer able to keep up with the increasing complexity of data. Such algorithms are the best choice when the data size is limited and well-formed in tabular format. Previously, researchers were interested in analyzing structured data inserted by experts into databases, but now scientists use machine learning in various aspects of life. Most data is unstructured, such as images, text, speech, and videos. In addition, the volume of data has significantly increased, and traditional machine learning algorithms cannot handle these types of data at the performance level expected by researchers. Artificial Neural Networks (ANN) [12][13] have experienced several waves of popularity and disillusionment, dating back to 1943. In recent years, with the improvement in computational power and data availability, the success and advancements in artificial neural networks and deep learning have been hot topics at machine learning conferences. In many problem domains, deep learning has achieved or surpassed human performance, making it a leading approach in various fields, from image processing and object detection to natural language processing and speech recognition.

To apply standard deep learning architectures to NLP, text data needs to be processed. After tokenizing the text data using available tools like NLTK, the next step is to convert the tokens (also known as words) into a suitable numeric format. There are multiple methods to achieve this, such as one-hot encoders or n-gram representations, but the most commonly used approach so far is using the Word2Vec algorithm. This creates custom-length vectors that represent the semantics of specific words and attempt to capture relationships between words. Word2Vec mappings can be trained from scratch, but it is also common to use a pre-trained word representation like GloVe as a feasible alternative.

In the field of computer science and artificial intelligence, natural language serves as a tool and effective means of interaction between humans and computers, playing an essential role in human communication. With the continuous advancement of AI technology, significant achievements have been made in natural language processing (NLP), and the capability to analyze the semantic meaning of text has become highly mature. In recent years, with the development of artificial intelligence, AI-based natural language processing (NLP) plagiarism detection methods have played an important role in text similarity calculation and plagiarism detection tasks. These methods utilize machine learning and deep learning techniques to analyze, compare, and measure the similarity between texts to

identify duplicated content or instances of plagiarism. AI-based NLP plagiarism detection methods have significant applications in academic research, document plagiarism detection, and essay evaluation. They can handle texts of varying lengths, consider semantic relationships between words and sentences, and are trained on large-scale datasets, resulting in high robustness and accuracy. Furthermore, with the continuous progress of deep learning and natural language processing technologies, AI-based NLP plagiarism detection methods will continue to improve and play a more significant role in practice.

In this paper, an AI-based approach is employed for plagiarism detection, combining statistical machine learning with deep learning techniques to design six models from three perspectives: character-level, word-level, and semantic-level. The six models are illustrated in Figure 1.

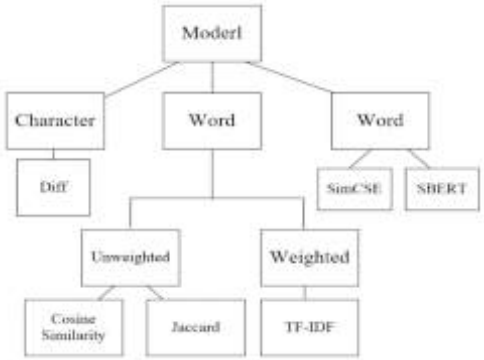

Fig. 1. From the perspectives of characters, words, and semantics, six models have been designed

2. Plagiarism Detection Models and Technical Approaches

2.1. The diff algorithm

It is a plagiarism detection algorithm based on string comparison, which measures the similarity between two texts by calculating the edit distance (i.e., the number of deletions, insertions, or replacements needed to transform one text into another).

Specific Examples are shown below:

a: Development and Implementation

b: Implementation Tasks

Similarity Score = 2 * len(T) / (len(a) + len(b)) = 0.5

2.2. Cosine similarity algorithm

The cosine similarity algorithm is a plagiarism detection method based on the vector space model. It measures the similarity between two texts by representing them as word frequency vectors and calculating the cosine value of the angle between the vectors.

Specific Examples are shown below:

a: Device | Alarm | Information |, | Device | Alarm | Level

b: Device | Communication | Status | Information |, | Device | Link | Information

Vocabulary: [Device, Alarm, Information, Level, Communication, Status, Link]

V_a: [1,1,1,1,0,0,0]

V_b: [1,0,1,0,1,1,1]

Similarity Score = cosine(V_a,V_b)=0.447

2.3. Jaccard algorithm

The Jaccard algorithm, another plagiarism detection approach based on the vector space model, measures the similarity between two texts by representing them as sets of words and calculating the ratio of the intersection to the union of the sets.

Specific Examples are shown below:

a: Device | Alarm | Information |, | Device | Alarm | Level

b: Device | Communication | Status | Information |, | Device | Link | Information

Vocabulary: [Device, Alarm, Information, Level, Communication, Status, Link]

V_a: [1,1,1,1,0,0,0]

V_b: [1,0,1,0,1,1,1]

Similarity Score = len(V_a& V_b) / len(V_a ｜ V_b)=2/7=0.286

2.4. TF-IDF algorithm

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical method used to evaluate the importance of a term in a collection of documents. In this method, TF refers to term frequency, which measures the term's ability to describe the content of a document, and IDF refers to inverse document frequency, which measures the term's ability to differentiate documents. The TF-IDF algorithm is based on the following assumptions: if a term appears frequently in a document, it is likely to appear frequently in other similar documents, and vice versa. Therefore, using the term frequency (TF) as a measure in the feature space coordinates can reflect the characteristics of similar texts. To account for the distinguishing power of words across different categories, TF-IDF introduces the concept of inverse document frequency (IDF), which suggests that the less frequently a word appears in the text corpus, the greater its ability to differentiate different categories of texts. Thus, the product of TF and IDF is used as the value measure in the feature space coordinates, adjusting the weight of TF to highlight important words and suppress less significant ones.

Specific Examples are shown below:

a: Device | Alarm | Information |, | Device | Alarm | Level

b: Device | Communication | Status | Information |, | Device | Link | Information

Vocabulary: [Information, Alarm, Alarm Information, Level, Device, Status, Communication, Link]

Weight = Term Frequency * Inverse Document Frequency

V_a：[0,0,0.816,0.408,0.408,0,0,0,0]

V_b：[0,0,0,0,0,0,0.577,0.577,0.577]

Similarity Score = cosine(V_a,V_b)=0

2.5. SimCSE algorithm

The SimCSE algorithm is a plagiarism detection method based on self-supervised learning. It obtains text representations by training a semantic matching task and then measures the similarity between two texts using cosine similarity.

Figure 2 illustrates the architecture of SimCSE, with the left side representing the unsupervised SimCSE and the right side representing the supervised SimCSE[14].

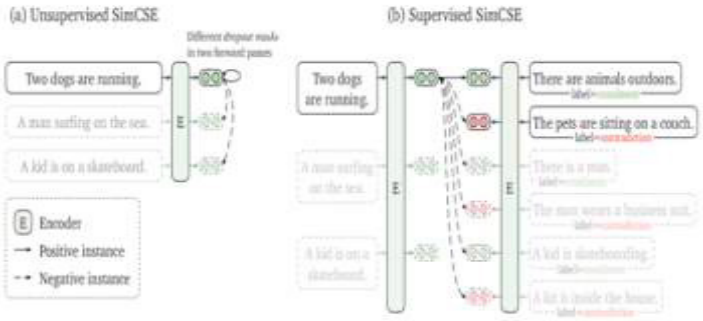

Fig. 2. SimCSE Structure

Similarity Score = cosine(V_a, V_b)
Training Loss: Cross-Entropy Loss

2.6. SBERT algorithm

The SBERT algorithm is a plagiarism detection method based on the Siamese network architecture. It maps texts into two separate embedding spaces and measures their similarity using cosine similarity. The SBERT model enhances its performance through various techniques such as multi-task training, data augmentation, and negative sample sampling[15].

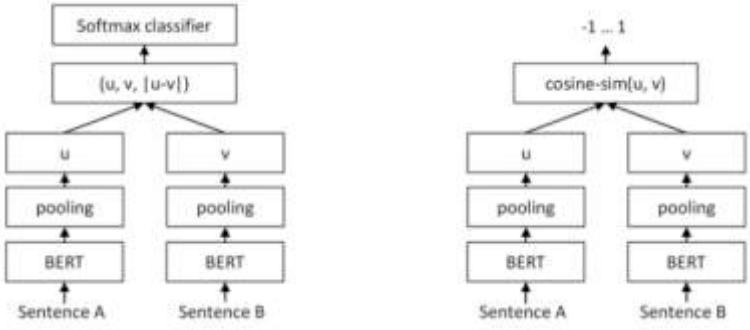

Fig. 3. SBERT Structure

Specific Examples are shown below:
Objective Similarity Score = (diff + cosine similarity + Jaccard similarity + tf-idf) / 4
Similarity Score = cosine(V_a, V_b)
Training Loss: Mean Squared Error
For multiple expressions of the same sentence, data augmentation is employed by translating the original Chinese sentence to Japanese, Russian, and English. The objective similarity score is enhanced by subtracting 0.1 or 0.2 from the original similarity score.

2.7. Technical approach

The technical approach involves computing similarity scores between the target project and all existing projects, sorting them based on the similarity scores, and returning the sorted results along with the corresponding similarity scores. To make full use of the characteristics of different models and improve the algorithm's robustness, a

combination of six models is used to compute the similarity scores. Based on the experience of Professor at Nanjing University and multiple experimental results, the specific methodology is designed as follows:

Initial similarity score = (Diff + Cosine Similarity + Jaccard + TF-IDF) / 4

Scenario 1:

When the initial similarity score is relatively low (initial similarity score <= 0.3), it indicates a low similarity between the target project and existing projects. In this case, the initial similarity score is directly used as the final similarity score to avoid excessive computation.

Final similarity score = Initial similarity score

Scenario 2:

When the initial similarity score falls between 0.3 and 0.6, it indicates a moderate similarity between the target project and existing projects. To improve the plagiarism detection effectiveness, a combination of two deep learning models (SimCSE and SBERT) is used in addition to the initial similarity score.

Final similarity score = (Diff + Cosine Similarity + Jaccard + TF-IDF + SBERT + SimCSE) / 6

Scenario 3:

When the initial similarity score is relatively high (greater than 0.6), it indicates a high similarity between the target project and existing projects. In this case, the TF-IDF model's similarity score is directly used as the final similarity score to avoid excessive computation and improve efficiency.

Final similarity score = TF-IDF

3. Experimental Data and Content

3.1. Experimental data

In this study, the data source consisted of a total of 1328 internship reports from the graduating students of the Department of Computer Science at a certain university during the period of 2020-2021. The total text size was 1,300,385 bytes. The reports covered various computer science internship projects, including topics such as "Software Engineering," "Web Design," and "System Design." To ensure the accuracy of the categories, the reports were manually annotated and divided into training and testing sets. Table 1 presents the sampling method used, which involved simple random sampling for data classification. By default, each category had 50 reports in the testing set, while the remaining articles were assigned to the training set.

Table 1. Experimental data categories and assignments

Categories	Training Set	Testing Set
Mobile App Design	282	50
Online System Design	302	50
Data Mining	262	50
Other	157	50

3.2. Experimental content

To ensure the recognition of the six models in terms of algorithm correctness, readability, robustness, and time-space complexity, this study primarily focuses on correctness and time-space complexity. The following procedures were designed for the experimental result comparison.

1) *Complete the preliminary preparations described in the algorithm design. Preprocess the data text from Table 1 to form an initial dataset. Subsequently, perform tasks such as word frequency ranking and word vectorization, removing words with low frequencies to create the final dataset.*
2) *Train the six models using the training set. Save the topic model as lda_model for comparison. During the training process, set the number of model topics to 50.*
3) *Optimize the training of the topic model. Save the topics extracted by the LDA model into a list. Then, find the corresponding weight values for these words under each topic. Save the product of the weight values*

and word vectors as importance weights, which serve as model parameters. Train the model again and save it as is_lda_model.

4) *Compare the topic categories and probability distributions of the two models' topic words. Calculate the training accuracy of the models.*

4. Experimental Results Analysis

4.1. Model training result comparison

Model training plays a crucial role in plagiarism detection algorithms. The classified training sets were separately input into the models for training, with the parameter settings of 5 topics for each category, and 10 topic words under each topic. Table 2 presents the similarity scores of the six models for the five topics.

Table 2. Analysis of data training results

	Diff	Cosine Similarity	Jaccard	TF-IDF	SimCSE	SBERT
System	0.429	0.458	0.498	0.486	0.499	0.487
Design	0.221	0.202	0.208	0.365	0.288	0.257
Testing	0.301	0.295	0.335	0.365	0.325	0.298
Process	0.245	0.265	0.266	0.245	0.358	0.369
Function	0.736	0.768	0.764	0.775	0.754	0.786

4.2. Comparing model accuracy results

When the initial similarity score falls between 0.3 and 0.6, it indicates a moderate similarity between the target project and existing projects. To enhance the accuracy of text similarity detection, a combination of two deep learning models, namely SimCSE and SBERT, was employed in addition to the initial similarity score. Therefore, similarity detection of known duplicate text was performed using four models (Diff + Cosine Similarity + Jaccard + TF-IDF) as well as six models (Diff + Cosine Similarity + Jaccard + TF-IDF + SBERT + SimCSE). The results are presented in the following Table 3.

Table 3. Algorithm Performance Comparison

	Diff + Cosine Similarity + Jaccard + TF-IDF	Diff + Cosine Similarity + Jaccard + TF-IDF + SBERT + SimCSE
0.4	0.389	0.397
0.5	0.493	0.501
0.6	0.588	0.597

We employed the Mean Absolute Error (MAE) to quantify accuracy. The MAE represents the average absolute difference between observed values and true values, effectively measuring the average magnitude of errors between predicted values and true values. For the four models, the average absolute error was 0.01, while for the six models, it was 0.00167. Thus, we can conclude that the introduction of SimCSE and SBERT led to improved accuracy.

4.3. Algorithm performance comparison

Known similarity scores (0.9, 0.5, 0.1) were input into the models for training to assess the algorithm's performance and resource consumption by distinguishing between three different scenarios. The results are presented in Table 4.

Table 4. Algorithm Performance Comparison

	Diff	C-S	Ja	TF-IDF	SimCSE	SBERT	Time Consumption / Memory Usage	Actual Time Consumption / Memory Usage

0.9	0.861	0.895	0.894	0.899	0.901	0.903	2.96（3～5%）	0.899/2.62/(2%)
0.5	0.485	0.492	0.495	0.497	0.501	0.497	3.02（3～5%）	0.492/3.01(4%)
0.1	0.114	0.115	0.114	0.103	0.107	0.104	3.11（3～5%）	0.109/2.52/(2%)

4.4. Results analysis

All six models demonstrate outstanding effectiveness in text similarity detection. When the four models (Diff + Cosine Similarity + Jaccard + TF-IDF) are combined with SBERT + SimCSE, the accuracy of text similarity detection is improved by a factor of 10. Furthermore, in practical applications, considering scenario two, the addition of SBERT + SimCSE enhances both the utilization of computational resources and the accuracy of text similarity detection. However, in scenario one and scenario three, where SBERT + SimCSE models are not utilized, the detection time is reduced and the resource utilization decreases.

5. Conclusions and Suggestions

This paper presents an optimized algorithm design for Text similarity detection based on artificial intelligence (AI) and natural language processing (NLP). The importance of NLP in computer science for detecting plagiarism in texts is highlighted. The paper provides a detailed overview of the experimental data and result analysis, exploring the performance of different models under various scenarios. In the experimental section, six different models were employed for plagiarism detection and comprehensively analyzed. The results demonstrate excellent performance of all six models in data detection. By leveraging the strengths of these models and analyzing their effectiveness in three different scenarios, the algorithm's robustness is enhanced, achieving remarkable plagiarism detection time and resource utilization. Overall, the proposed method for Text similarity detection, based on AI and NLP, exhibits high accuracy and robustness. These methods can handle texts of different lengths and consider semantic relationships between words and sentences. Moreover, they can be trained on large-scale datasets to achieve higher accuracy. Furthermore, as deep learning and NLP technologies continue to advance, AI and NLP-based Text similarity detection methods will continue to improve and play a more significant role in practice. Future research can explore additional models and algorithms to further enhance the accuracy and robustness of plagiarism detection.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[2] A. Kamilaris and F. X. Prenafeta-Boldú, “Deep learning in agriculture: A survey,” Comput. Electron. Agric., vol. 147, pp. 70–90, 2018.

[3] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. MIT press, 2016.

[4] D. Learning, “Deep learning,” High-dimensional fuzzy Clust., 2020.

[5] H. Tian, T. Wang, Y. Liu, X. Qiao, and Y. Li, “Computer vision technology in agricultural automation—A review,” Inf. Process. Agric., vol. 7, no. 1, pp. 1–19, 2020.

[6] S. Xu, J. Wang, W. Shou, T. Ngo, A.-M. Sadick, and X. Wang, “Computer vision techniques in construction: a critical review,” Arch. Comput. Methods Eng., vol. 28, pp. 3383–3397, 2021.

[7] D. W. Otter, J. R. Medina, and J. K. Kalita, “A survey of the usages of deep learning for natural language processing,” IEEE Trans. neural networks Learn. Syst., vol. 32, no. 2, pp. 604–624, 2020.

[8] Y. Kang, Z. Cai, C.-W. Tan, Q. Huang, and H. Liu, “Natural language processing (NLP) in management research: A literature review,” J. Manag. Anal., vol. 7, no. 2, pp. 139–172, 2020.

[9] M. A. Chandra and S. S. Bedi, “Survey on SVM and their application in image classification,” Int. J. Inf. Technol., vol. 13, pp. 1–11, 2021.

[10] M. P. LaValley, “Logistic regression,” Circulation, vol. 117, no. 18, pp. 2395–2399, 2008.

[11] D. Chandrasekaran and V. Mago, “Evolution of semantic similarity—a survey,” ACM Comput. Surv., vol. 54, no. 2, pp. 1–37, 2021.

[12] B. Yegnanarayana, Artificial neural networks. PHI Learning Pvt. Ltd., 2009.

[13] A. K. Jain, J. Mao, and K. M. Mohiuddin, “Artificial neural networks: A tutorial,” Computer (Long. Beach. Calif)., vol. 29, no. 3, pp. 31–44, 1996.

[14] T. Gao, X. Yao, and D. Chen, “SimCSE: Simple Contrastive Learning of Sentence Embeddings,” 2021.

[15] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” 2019.