Research Article

Context-Aware Text Matching Algorithm for Korean Peninsula Language Knowledge Base Based on Density Clustering

Li Xiang^{1,2} and Li ZongXun³

¹College of Humanities and Social Sciences, Yanbian University, Yanji, Jilin 133002, China

²School of Literature and History, Mianyang Teachers' College, Mianyang 621000, Sichuan Province, China

³Journal Editorial Department, Yanbian University, Yanji, Jilin 133002, China

Correspondence should be addressed to Li Xiang; 160701132@stu.cuz.edu.cn

Received 17 August 2021; Accepted 28 September 2021; Published 7 October 2021

Academic Editor: Fazlullah Khan

Copyright © 2021 Li Xiang and Li ZongXun. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The majority of the traditional methods deal with text matching at the word level which remains uncertain as the text semantic features are ignored. This also leads to the problems of low recall and high space utilization of text matching while the comprehensiveness of matching results is poor. The resultant method, thus, cannot process long text and short text simultaneously. The current study proposes a text matching algorithm for Korean Peninsula language knowledge base based on density clustering. Using the deep multiview semantic document representation model, the semantic vector of the text to be matched is captured for semantic dependency which is utilized to extract the text semantic features. As per the feature extraction outcomes, the text similarity is calculated by subtree matching method, and a semantic classification model based on SWEM and pseudo-twin network is designed for semantic text classification. Finally, the text matching of Korean Peninsula language knowledge base is carried out by applying density clustering algorithm. Experimental results show that the proposed method has high matching recall rate with low space requirements and can effectively match long and short texts concurrently.

1. Introduction

With the rapid development of the digital society, people's needs in the fields of artificial intelligence such as information retrieval, automatic question answering, and dialogue systems have begun to appear, and intelligent matching algorithms are needed to meet the high needs of the users [1]. In order to meet these requirements, natural language processing technology emerged, which can provide users with efficient information retrieval services [2]. Text matching algorithm is the core research area in natural language processing technology. The dimension disaster and sparsity of data in the traditional text matching field have affected the development of natural language processing. Moreover, the majority of the traditional text matching models ignore the relationship between words and cannot recognize the semantic similarity between words [3]. To address the listed problems, researchers have proposed multiple text matching methods using modern technologies.

Chen et al. combined the idea of transforming trie tree into double array form by proposing an improved multi-pattern matching algorithm based on Aho-Corasick algorithm [4]. The method is based on a string searching mechanism that locates elements of a finite set of strings within the input string. A finite state machine is constructed resembling a trie having additional links between various internal nodes which allow the automation of transition between string matches without needing any backtracking. According to the analysis of the results of comparative experiments, the algorithm not only successfully matched all the pattern strings to be found in the text, but also had a fast processing speed. On the other hand, Wu proposed a text matching method combining pretraining model and language knowledge base [5]. Based on large-scale pretraining model, this method introduced external linguistic knowledge by generating synonym antonym vocabulary learning task and phrase collocation learning task via utilizing WordNet jointly trained with Multi-Task Deep Neural

Network (MT-DNN) to further improve the performance of the model. In the end, text matching annotation data are used for fine-tuning. The experimental results on two public data sets, Microsoft Paraphrase Corpus (MRPC) and Quora Question Pairs (QQP), testify that this method can effectively improve the performance of text matching by introducing external language knowledge for joint training on the basis of large-scale pretraining model and fine-tuning framework. Zen and Chen proposed a text matching model [6] based on word embedding and dependency via constructing a semantic representation obtained by integrating word meaning and dependency between words. The approach formulates a matrix describing the semantic matching degree of each part of the two texts through cosine mean convolution and K-MAX pooling operation and then uses long-term and short-term memory network to learn the mapping relationship between the matching degree matrix and the real matching degree. The experimental results show that the text matching accuracy of the model is reasonably high. In continuation, Xu et al. [7] proposed a deep learning model based on self-learning text utilizing nearest neighbor graph framework to deal with short text matching. The nearest neighbor graph can use word embedding to convert the text into vector form. The nearest neighbor relationship is formulated on the vectors to represent the text samples by constructing the text similarity relationship matrix. Twin convolution neural network is used afterwards to learn a better nearest neighbor graph in order to complete the text matching task. Experimental results reveal that the model can effectively improve the accuracy of text recognition and matching.

Although the discussed methods solve text matching problem to a certain extent, these methods mostly deal with text matching problems at the word level and do not consider the semantic level of the text, resulting in unsatisfactory text matching effects and low matching recall rates. This also results in the problem of high space utilization, and the comprehensiveness of the method is not much good. Moreover, these are not applicable to the processing of long text and short text at the same time. As a better alternative, this paper proposes a text matching algorithm based on density clustering in Korean Peninsula language knowledge base to solve the problems of traditional methods and to improve the overall text matching effect.

2. Text Matching Algorithm of Korean Peninsula Language Knowledge Base

The proposed method is divided into four steps for better explanation and understanding as explained in the following sections.

2.1. Text Semantic Feature Extraction. As the first step, the deep multiview semantic document representation model is utilized to capture the semantic dependency of the semantic vector of the matched text. Equation (1) represents the process of capturing the semantic dependency of the text for a text sequence $A = \{a_1, a_2, \dots, a_n\}$:

$$A(t) = P \int \frac{a(t)}{(1-t')}, \quad (1)$$

where P represents the semantic vector of the text, $a(t)$ represents the semantically similar text segment, and t' represents the similar text with the interference information added.

Once the semantic dependency is captured, it is necessary to perform feature extraction and feature selection of high-dimensional abstract semantic information for word granularity and sentence granularity [8]. A deep neural network is used to construct a multigranularity semantic feature extraction model as shown in Figure 1.

As described in Figure 1, the convolution layer of the deep neural network uses single dimensional convolution kernel for feature extraction of high-dimensional semantic information at the word granularity level as presented in

$$x(t) = \frac{1}{N} \int a^2 W_x(a, b) \psi_{a,b}, \quad (2)$$

where $x(t)$ represent the output semantic vector of the convolution layer, N represent the total text, a^2 represent the bias parameter of the convolution kernel, W_x represent the convolution operation, and $\psi_{a,b}$ represent the size of the shared weight of the convolution kernel, considering a as the length of the convolution kernel and b as the width of the convolution kernel.

In the succeeding step, the in-depth multiview semantic document representation model adopts the global maximum strategy in the pooling layer to perform feature selection for the high-dimensional semantics at the sentence granularity level and extract the most important semantic features as shown in

$$f(t) = \sum_{i=1}^n A_i(t) \omega_j(t) dt, \quad (3)$$

where $f(t)$ represent the semantic features outputted by the pooling layer, $A_i(t)$ represent the candidate text vector set, and $\omega_j(t)$ represents the dependency between semantics.

After feature extraction and feature selection of semantic information at the levels of word granularity and sentence granularity by convolution neural network, the text to be matched becomes a new high-dimensional abstract semantic vector. In the subsequent steps, the resultant dense vector to be matched is further processed by using the interaction function to obtain the text semantic features.

2.2. Text Similarity Calculation Based on Subtree Matching. An interesting way to find the similarity between related fragments of text is via trie matching. Although multiple solutions are available in the literature, but majority are derived from two basic techniques, i.e., Tree-to-Tree similarity matching and subtree similarity matching where the former matches the subject trees and returns a measure of their distance. The cost of calculating this function varies, ranging from linear time algorithms to algorithms which solve NP-hard problems. Considering long sequences of text

FIGURE 1: Multigranularity semantic feature extraction model.

as addressed in current study, the tree-to-tree similarity is not much suitable to be adopted as a potential solution. On the other hand, “subtree similarity matching” [9] finds all the subtrees among the subjects which are most similar to each other. This option is potentially scalable and could be successfully used for both short and long sequences of text as per the requirements of the current study. So as per the semantic features of the text obtained by the aforementioned calculation, the subtree matching method is used to calculate the text similarity. Since the matching subtree of each text is often different, so while designing a text similarity algorithm for different texts s_a and s_b , it is necessary to consider that the two may have the same matching subtree or they may have different matches. In the following text both possible cases are discussed separately.

2.2.1. When Text s_a and Text s_b Have the Same Matching Subtree. Ideally, when text s_a and text s_b have the same matching subtree T_j , the matching subtree is used as an intermediary and the text metadata feature vectors of both texts will have the highest degree of semantic overlap. Since the text metadata feature vector can characterize the text, the similarity between the two texts will be obviously high. The similarity relationship between texts based on the same matching subtree is shown in Figure 2.

In Figure 2, the matching subtree acts as an intermediary bridge for the calculation of similarity among the two text samples under discussion. Similarity 1 and similarity 2 represent the similarity between text s_a and text s_b and the matching subtree T respectively, which can be calculated by equation (4). Similarity 3 is the similarity between text s_a and text s_b calculated by using the matching subtree as the intermediary, which can be calculated by equation (5):

FIGURE 2: Schematic diagram of text similarity relationship under the same matching subtree.

$$\text{Sim}(s_a, s_b) = [W_x(a, b)]^2 d_i, \quad (4)$$

$$\text{Sim}(s_a, s_b) = [W_y(a, b)]^2 d_j, \quad (5)$$

where $\text{Sim}(s_a, s_b)$ represents the similarity between text s_a and text s_b ; W_x and W_y both represent the average similarity between text and the matching subtree, respectively, while d_i and d_j both represent the difference between text s_a and text s_b .

While calculating whether two texts are similar, it is necessary to judge the influence of the difference between the text and the matching subtree similarity on overall text similarity. If absolute value of the difference is large, the text similarity value will eventually decrease [10]. Equation (6) describes the formula for the calculation of similarity difference between the text and the matching subtree:

$$\text{Sim}\sigma_i = \min \frac{\|b_i - x_i^{(k)}\|}{\|b_j - x_j^{(k-1)}\|}, \quad (6)$$

where b_i and b_j both represent the local relevance of the text, $x_i^{(k)}$ represents the joint feature vector, and $x_j^{(k-1)}$ represents the interactive information between the texts. It could be seen that a larger value of $\text{Sim}\sigma_i$ will result in a slower value of similarity between texts.

2.2.2. When Text s_a and Text s_b Have Different Matching Subtrees. It is usually a special case that two texts have the same matching subtree. More often, text s_a and text s_b have different matching subtrees. The similarity relationship between text s_a and text s_b is shown in Figure 3.

In Figure 3, subtree T_a is the matching subtree of text s_a and subtree T_b is the matching subtree of s_b . The matching subtrees T_a and T_b act as an intermediary bridge when calculating the similarity of the two texts where similarity 1 represents the similarity $\text{Sim}(s_a, T_a)$ between text s_a and subtree T_a , and similarity 2 represents the similarity $\text{Sim}(T_a, T_b)$ between subtree T_a and subtree T_b . Similarly, similarity 3 represents the similarity $\text{Sim}(s_b, T_b)$ between the text s_b and the subtree T_b . When the three similarities are all known, the similarity between text s_a and text s_b can be computed by

$$\text{Sim}(s_a, s_b) = \frac{R_{ab} \times H_{ab}}{2}, \quad (7)$$

FIGURE 3: Schematic diagram of text similarity relationship in different matching subtrees.

where R_{ab} represents the number of valid words in the database and H_{ab} represents the length of the original text.

Similar to the first case, it is also required to determine the impact of the difference between the three similarities on text s_a and text s_b as expressed in

$$\text{Sim}\sigma_{ij} = \min\left(\frac{\partial u_i}{\partial x_i} + \frac{\partial u_j}{\partial x_j}\right) \times \text{Sim}(s_a, T_a) + \min\left(\frac{\partial u_i}{\partial x_i} + \frac{\partial u_j}{\partial x_j}\right) \times \text{Sim}(T_a, T_b) + \min\left(\frac{\partial u_i}{\partial x_i} + \frac{\partial u_j}{\partial x_j}\right) \times \text{Sim}(s_b, T_b), \quad (8)$$

where u_i and u_j both represent the high-dimensional abstract text semantic vector and ∂ represent the matching function.

So, in a nutshell, the similarity calculation process of the text s_a and text s_b for different matching subtrees mainly includes the following three steps:

- Step 1: calculate the similarity between text s_a and text s_b and their matching subtrees T_a and T_b , respectively
- Step 2: calculate the similarity between the matching subtrees T_a and T_b
- Step 3: use matching subtrees T_a and T_b as an intermediary to calculate the similarity between text s_a and text s_b

2.3. Semantic Classification Model Based on SWEM and Pseudo-Twin Network. Word embedding is a language modeling technique that represents words or phrases as real-number vectors. The words are grouped together to create a representation that is comparable to that of words with similar meanings. To construct the representation, word embedding learns the relationship between the words. Several methods including probabilistic modeling, cooccurrence matrix, and neural network based methods could be utilized for the calculation of word embedding.

Simple Word Embedding-based Model (SWEM) [11] is a model based on word vectors that utilize pooling technology. The module itself has no parameters. The twin network constructed with this model has fewer parameters and can be trained faster. The semantic classification model constituted upon SWEM mainly includes input layer, SWEM layer, aggregation layer, and output layer. The input layer is a word vector, which can be fine-tuned or fixed directly according to the training process. The SWEM layer uses a pooling method to process the text, the aggregation layer uses a common distance measurement algorithm to obtain the distance between the text vectors represented by SWEM, and the output layer is a simple two-class classification system to determine whether the texts are similar.

For the subsequent explanation of SWEM, suppose a text pair $B = \{b_1, b_2, \dots, b_{L_1}\}$ with length L_1 and $D = \{d_1, d_2, \dots, d_{L_2}\}$ with length L_2 . Similarly consider the set of labels $C \in [0, 1]$, where 0 means that the text is not similar and 1 means that the text is similar.

2.3.1. Input Layer. The text is directly segmented into single words, and Word2Vec is used to train word vectors for all text. During the training process, it is found that keeping the word embedding layer trainable may cause the model to overfit, so the trainable parameter is set to “No,” which actually reduces the difference in word vectors between the training set and the test set [12]. The two texts are represented as $V_b = R^{L_1 \times z}$ and $V_d = R^{L_2 \times z}$ through the above-mentioned operations where z is the size of the word vector. In order to ensure that the word vector can be integrated, the word vector size z is set to 300 in the pretraining stage.

2.3.2. SWEM Layer. This layer uses three variants of SWEM mainly including two pooling technologies and a fusion method: SWEM-max, SWEM-aver, and SWEM-concat, all of which are explained in the following text:

- (1) SWEM-aver: it averages the word vector by element which is equivalent to average pooling. This method uses each element of the word vector that is equivalent to fusing the information of each word. The advantage of SWEM-aver is that the information of each sequence element can be considered in the result through the addition operation, which can be expressed by the formula mentioned in

$$P_k^b = u_{\text{eff}} \frac{\partial b}{\partial x_i}, \quad (9)$$

$$P_k^d = u_{\text{eff}} \frac{\partial d}{\partial x_i}, \quad (10)$$

where P_k^b and P_k^d , respectively, represent the input text vector sequence and the candidate text vector sequence while u_{eff} represents the matching probability likelihood function.

- (2) SWEM-Max: this variant is equivalent to the maximum pooling technology, SWEM-max brings very good interpretability to the model as the text vector trained with SWEM-max has great sparsity. From the perspective of specific tasks, the words that can highlight the theme in each text can now be easily selected.
- (3) SWEM-concat: it combines average pooling and maximum pooling by complementing the two pooling methods and splice the results obtained by the above two pooling methods.

For all the three SWEM variant models, no internal components of the model need to learn explicitly as these models only use inherent word embedding information for text classification, which can minimize the semantic classification time and improve the classification efficiency.

2.3.3. Aggregation Layer. The most important function of the aggregation layer is to aggregate the (two) obtained text representations into a fixed-length matching vector which is carried out using distance measurement formulas or fixed splicing methods for aggregation. The distance measurement formula adopts the most common splicing formula in the twin network. It is the specialty of the twin network to utilize the target values of two different data points (in this case “vectors”) rather than the targets themselves. For this purpose, the two text vectors are multiplied and subtracted to obtain the original vector of the two texts. Finally, a long vector is obtained as the aggregation vector as presented in

$$P_{\text{fin}} = \frac{J_i + R_i + S_i}{\rho Y_i Y_j}, \quad (11)$$

where R_i represent the matching score function while S_i represent the multigranularity matching information.

2.3.4. Output Layer. Since the text labels are presented as 0 and 1, therefore, the output of the text classification result is fed into second classification. The result obtained by the distance measurement layer is passed through a fully connected layer and then the final text classification result is obtained through sigmoid [13] as per the following equation:

$$\vartheta_{ab} = \text{sigmoid}(\omega_{ab})^t. \quad (12)$$

2.4. Implementation of Text Matching in Korean Peninsula Language Knowledge Base Based on Density Clustering. Based on the results of text feature collection, text similarity calculation, and text classification processing, the density clustering algorithm is used to match the Korean peninsula language knowledge base text [14]. The density clustering

algorithm is an unsupervised clustering algorithm based on high-density connected areas. In the entire sample space, each target cluster is composed of a group of dense sample points divided by low-density areas. The purpose of the algorithm is to filter low-density areas and find dense sample point.

For the current study, a sample space is set, and the text having actual semantics is distributed along the diagonal direction of the cluster interval. Therefore, the slope density of the defined cluster interval is ω_u which is calculated by the following formula:

$$\omega_u = \left(\frac{m_1 + m_2}{2} \right) \sin \omega_a \cos \phi_b, \quad (13)$$

where m_1 and m_2 represent the sum of projections of the matching points in the cluster on the corresponding coordinate axis; ω_a represents the cluster interval of the text s_a , while ϕ_b represents the cluster interval of the text s_b .

Moreover, the class cluster interval satisfies the conditions mentioned in

$$\begin{cases} \forall p_i \in \omega_a, & i = 1, 2, \dots, n, \\ \forall p_j \in \phi_b, & i = 1, 2, \dots, n. \end{cases} \quad (14)$$

It is said that the cluster interval ω_a contains ϕ_b , which is referred to as “cluster contents” and is denoted as $\phi_b \subseteq \omega_a$.

Similarly, the clusters meet the conditions of

$$\begin{cases} \exists p_i \in \omega_a, & i = 1, 2, \dots, n, \\ \exists p_j \in \phi_b, & i = 1, 2, \dots, n. \end{cases} \quad (15)$$

It is said that the cluster intervals ω_a and ϕ_b intersect referred to as “cluster intersection” and are denoted as $\phi_b \cap \omega_a$.

According to the abovementioned analysis, each matching point is initialized to a cluster before text matching, and the set containing all clusters is set to E . According to the related definition of density clustering, the matching rules are set as follows:

Rule 1: if the density of two clusters is reachable, merge the two clusters, recalculate the cluster interval of the new cluster, and delete the original cluster from E .

Rule 2: if two clusters meet the cluster inclusion conditions, delete the included clusters.

Rule 3: exit condition, and traverse E ; no clusters can be merged.

According to Rule 3, if clusters are merged, the new clusters must be compared with the existing clusters again until there are no clusters that can be merged.

Based on the above analysis, it can be seen that the problem under discussion of current study can be implemented in two steps:

- (1) Order matching: according to the semantic characteristics of the text, the possibility of merging adjacent clusters is high. Therefore, first sort all the

initial cluster intervals in order, and then match according to Rule 1.

- (2) Iterative merging: on the basis of sequence matching, iterative merging is performed according to Rules 1 and 2 until no new merges are generated. The algorithm description is shown in Figure 4.

According to Figure 4, the postprocessing is to process the matching results according to a certain rejection condition where those clusters are discarded that do not meet the semantics. User can process either the merged clusters or the restored text fragments although, in general, it is better to process the restored text fragments [15]. According to the semantic characteristics of similar texts, the conditions for discarding clusters are defined as follows:

Condition 1: the cluster density exceeds the range set by the density deviation

Condition 2: the density of clusters is less than a preset threshold

The abovementioned conditions must be considered comprehensively. If the cluster density exceeds the density deviation, but its density is higher than a certain threshold, then it can still be considered as a candidate in the similar text matching problem.

3. Results

In order to verify the comprehensiveness and effectiveness of the proposed study and to prove its acceptability, various simulation experiments are carried out. In the experiments, the improved multimode matching algorithm based on Ah-Corasick algorithm [4] and the text matching method combined with pretraining model [5] are selected for comparison purpose. The matching recall percentage, space utilization, and comprehensiveness are used as comparison indexes for the comparative analysis.

3.1. Dataset and Evaluation Criteria. The experiment selects SQL dataset as the basic dataset and carries out data analysis through the MATLAB simulation software. The average number of words of all articles in the data set is 734, and the maximum number of words is 21791. Due to the influence of various factors in the experimental process, the experimental data can have errors. In order to overcome the impact of errors on the experimental results, repeated measurements are carried out to obtain the average value. Figure 5 is the output interface diagram of experimental results produced by MATLAB.

3.2. Analysis of Specific Results

3.2.1. Recall Rate. Considering the matching recall rate as the primary experimental index, different methods are compared, and the results are presented in Figure 6.

The higher the matching recall rate, the more comprehensive the matching results. As could be noticed in

FIGURE 4: Flowchart of density clustering algorithm implementation.

Figure 6, the text matching recall obtained by the proposed algorithm gets improved with the increasing number of iterations as the recall rate shows an obvious upward trend. Compared with the traditional methods, the text matching recall shows significant improvement. It proves that when the proposed algorithm is used to process similar text, more text can be matched as compared to other counterparts developed for the purpose. This is because before text matching, the algorithm utilizes the deep multiview semantic document representation model to capture the semantic dependency of the semantic vector of the text samples thereby extracting the semantic features of the text and thus the text can be matched more pertinently according to the extraction results.

3.2.2. Space Utilization. The second important measure to evaluate the experimental index is the space utilization rate. Figure 7 presents the results of three methods under discussion as per their utilized space.

FIGURE 5: Experimental output interface.

FIGURE 6: Comparison of text matching recall rates.

Lower space utilization means less space is occupied by the text matching method which makes it a more attractive choice. It could be noticed from Figure 7 that as the number of iterations increase, the space occupied by different methods for text matching shows an increasing trend. Compared with the traditional methods, the proposed algorithm utilized minimum space. This is because this algorithm uses the subtree matching method to calculate the text similarity according to the feature extraction results and the semantic classification model is designed based upon SWEM and pseudo-twin network to classify the text semantics. In this step, different types of text can be classified.

FIGURE 7: Comparison of space utilization.

The text is divided, and the text that does not belong to the same type is eliminated, which reduces the space occupied by invalid text.

3.2.3. Comprehensiveness of Matching Results. In order to further test the superior performance of the proposed algorithm, short and long text matching experiments are carried out to compare the matching accuracy of different methods. The test results are shown in Tables 1 and 2.

TABLE 1: Short text matching results.

Number of iterations/times	The proposed algorithm	Chen et al. [4]	Zhou et al. [5]
2	91.3	72.1	69.9
4	92.0	72.5	70.5
6	93.4	72.9	71.3
8	93.9	73.0	72.4
10	94.5	73.2	72.6
12	95.7	73.6	73.6

TABLE 2: Long text matching results.

Number of iterations/times	The proposed algorithm	Chen et al. [4]	Zhou et al. [5]
2	89.7	68.7	67.2
4	89.9	68.9	67.5
6	90.2	67.0	67.9
8	90.6	67.6	68.2
10	90.7	68.0	68.6
12	91.5	68.2	68.9

Analyzing the data in Tables 1 and 2, it can be seen that the text matching accuracy of the proposed algorithm is higher than that of the traditional method, whether it is short text or long text. Among them, the highest matching accuracy of the proposed algorithm for short text is 95.7%, whereas the highest matching accuracy rate for long text is 91.5%. These results indicate that the text matching results of the proposed algorithm are more generalized and comprehensive and could be used to obtain more useful text information.

4. Conclusion

The traditional text matching methods usually rely on word matching algorithms while ignoring the important information embedded into semantics of the sentence. In order to overcome the problem of poor text matching effect and to enhance the text matching results, a text matching algorithm based on density clustering in Korean Peninsula language knowledge base is proposed which has the following salient features:

- (1) The proposed system uses subtree matching to calculate text similarity. The similarity calculated among vectors in the semantic space can accurately describe the semantic relationship between different texts.
- (2) The characteristics of the natural language of human society include hierarchical structure, serialized structure, and combination operations. These can be well combined with the hierarchical and serialized characteristics of the deep text matching model itself.
- (3) The density clustering algorithm can make perfect use of large-scale data and the computing power of high-performance computers, starting from the laws of human natural language and improving the accuracy and effectiveness of the text matching.

Overall the results computed upon the above characteristics of the proposed model testify the superiority of the method over existing approaches. Hence it could be claimed that the proposed method may be adopted for practical

applications of Korean Peninsula language knowledge base text matching and could be further applied in other similar domains.

5. Future Direction

The context of a word with respect to its position in a phrase or sentence provides an important clue for its correct translation as demonstrated in the current study. Ambiguity is a situation when a word can be translated into two or more options and the system is confused upon the selection of the correct translation. In such a situation, the context can help in resolving the ambiguity. As a future direction for the researchers working in the area, the current study may be tested for ambiguous phrases and there are good possibilities that it could yield promising results.

The current study is developed by using density clustering; however, considering the number of advanced artificial intelligence tools developed by using machine learning, neural networks, and deep learning, the study may be improved by augmenting more modern AI methods.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this study.

References

- [1] A. Amir, T. Kopelowitz, A. Levy, S. Pettie, E. Porat, and B. R. Shalom, "Online dictionary matching with one gap," *Algorithmica*, vol. 81, no. 6, pp. 2123–2157, 2019.
- [2] J. Fize, M. Roche, and M. Teisseire, "Could spatial features help the matching of textual data?" *Intelligent Data Analysis*, vol. 24, no. 5, pp. 1043–1064, 2020.

- [3] D. Zheng, Y. Zhang, and Z. Xiao, "Deep learning-driven gaussian modeling and improved motion detection algorithm of the three-frame difference method," *Mobile Information Systems*, vol. 2021, Article ID 9976623, 2021.
- [4] Y. J. Chen, S. Wushour, and Q. Yu, "An improved multi-pattern matching algorithm based on Aho-Corasick algorithm," *Modern Electronics Technique*, vol. 42, no. 04, pp. 97–101, 2019.
- [5] Y. Wu, "Application of improved boosting algorithm for art image classification," *Scientific Programming*, vol. 2021, Article ID 3480414, 2021.
- [6] Z. Zhen and Y. Q. Chen, "Text matching model based on dependency relation and neural network," *Computer Engineering*, vol. 503, no. 08, pp. 216–222+229, 2019.
- [7] Y. Xu, F. Jiang, J. Du, and D. Gong, "A cross-domain collaborative filtering algorithm with expanding user and item features via the latent factor space of auxiliary domains," *Pattern Recognition*, vol. 94, pp. 96–109, 2019.
- [8] Q. Shi and J. Han, "Semantic feature extraction based on subspace learning with temporal constraints for acoustic event recognition," *Digital Signal Processing*, vol. 110, no. 7, Article ID 102947, 2020.
- [9] M. Yu, T. Quan, Q. Peng, Xu Yu, and L. Liu, "A model-based collaborate filtering algorithm based on stacked AutoEncoder," *Neural Computing and Applications*, vol. 5, pp. 1–9, 2021.
- [10] Q. Xu and J. Yin, "Application of random forest algorithm in physical education," *Scientific Programming*, vol. 2021, Article ID 1996904, 2021.
- [11] Y. Xiao, "Research on visual image texture rendering for artistic aided design," *Scientific Programming*, vol. 2021, Article ID 1190912, 2021.
- [12] M. Zhang, "Applications of deep learning in news text classification," *Scientific Programming*, vol. 2021, Article ID 6095354, 2021.
- [13] X. Yu, J. Yang, and Z. Xie, "Training SVMs on a bound vectors set based on fisher projection," *Frontiers of Computer Science*, vol. 8, no. 5, pp. 793–806, 2014.
- [14] E. Baracchini, L. Benussi, S. Bianco, C. Capoccia, and S. Tomassini, "A density-based clustering algorithm for the CYGNO data analysis," *Journal of Instrumentation*, vol. 15, no. 12, Article ID 12003, 2020.
- [15] L. Z. Feng, G. J. Yang, X. Xu, and Y. H. Xu, "Bi-direction matching Chinese word segmentation based on N-gram statistical model," *Jouranl of Applied Statistics and Management*, vol. 39, no. 4, pp. 633–643, 2020.