Short Text Similarity Calculation Based on Jaccard and Semantic Mixture

Shushu Wu[1(✉)], Fang Liu[1], and Kai Zhang[1,2]

[1] School of Computer Science, Wuhan University of Science and Technology, Wuhan 430081, China
[2] Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan 430081, China

Abstract. For the sake of enhancing the accuracy of short text similarity calculation, a short text similarity calculation method on account of Jaccard and semantic mixture is proposed. Jaccard is a traditional similarity algorithm based on literal matching. It only considers word form, and its semantic calculation has certain limitations. The word vector can represent the semantic similarity by computing the cosine similarity of two terms in the vector space, and the semantic similarity is obtained by adding and averaging the word similarity of two sentences according to a certain method. The two methods are now weighted to compute the final text similarity. Experiments show that the algorithm improves the recall rate and F value of short text calculation to some extent.

Keywords: Short text similarity · Jaccard · Word vector · Semantic similarity

1 Introduction

In the wake of the development of computer technology and the Internet, more and more information is presented in short texts. How to correctly compute the similarity of short texts has become particularly important, and it has also become a hot spot in natural language processing. Text similarity refers to the degree of semantic similarity of text. It can not only be applied to search engines, document duplicate checking, and automatic question and answer systems, but also can be applied to document classification and clustering and accurate document push [1]. We have brought great convenience. Compared with long texts, short texts have shorter content and sparse words, which makes calculations more difficult. For example, the same words can express different meanings, and different words can also express the same meaning, that is, the so-called polysemous and multi-sense words. And even if the word composition of the two sentences is exactly the same, but their combined structure is different, the meanings expressed are also different. According to the characteristics of similarity calculation methods, text similarity can be divided into literal matching similarity, semantic similarity and structural similarity. The calculation method of literal matching only considers the similarity of the text from the morphology,

L. Pan et al. (Eds.): BIC-TA 2020, CCIS 1363, pp. 37–45, 2021.
https://doi.org/10.1007/978-981-16-1354-8_4

which has great limitations; the semantic similarity method solves the semantic matching of words, but it needs to rely on the corpus; the structural similarity can analyze the grammatical structure of the text, But the accuracy will decrease with the increase of sentence length [2]. The three calculation methods have their own advantages and disadvantages, and they all need certain optimization.

For the study of short text similarity, Huang Xianying et al. added word order to the term, and combined the overlap similarity algorithm with the word order similarity algorithm between common word blocks to calculate the short text similarity [3]. Gu Zhixiang et al. used part of speech and word frequency weighting to improve the simhash algorithm [4]. Li Lian et al. optimized the vector space model algorithm by considering the influence of the same feature words between texts on the text similarity [5]. These methods all consider other features of the term on the basis of the word shape, and improve the text similarity algorithm to a certain extent, but do not involve the semantic level of the sentence. The realization of sentence semantics can use corpus, such as HowNet, WordNet, etc. Yuan Xiaofeng uses HowNet to calculate the semantic similarity of words, and uses the TF/IDF values of a small number of feature words to assign weights to the vectors in the VSM, and then computes the similarity between texts [6]. On the basis of improving the edit distance, Che Wanxiang et al. used two semantic resources HowNet and synonym cilin to calculate the semantic distance between words, and obtained better results on the basis of taking into account word order and semantics [7]. Zhang Jinpeng et al. studied text similarity based on the vector space model and semantic dictionary, and discussed the semantic similarity of texts of different lengths and their applications [8]. Liao Zhifang et al. proposed a short text similarity algorithm based on syntax and semantics. By calculating the similarity of short texts with the same syntactic structure and considering the contribution of sentence phrase order to the similarity, the similarity of Chinese short texts was calculated [9]. These methods consider the part of speech and semantics of the sentence, but the semantics need to rely on an external dictionary, which cannot calculate the semantic similarity of words between different parts of speech. The word vector makes up for this shortcoming. It can also be used for extended training based on the corpus and the one-hot vector can be converted into a low-dimensional word vector. Therefore, Jaccard is combined with a semantic algorithm based on word vectors to find text similarity. Jaccard is an algorithm based on literal matching, which takes into account the morphology of the text. It is suitable for calculating two sentences with more co-occurring words, but for two sentences that do not overlap at all, the calculated similarity is 0 and cannot be calculated Similarity of similar words. The word vector can calculate the semantic similarity of words, which makes up for the shortcomings of the Jaccard algorithm, so the two are combined to find the short text similarity.

The first part of this article mainly introduces the short text and some related research content, the second part describes the algorithm used in detail, the third

part is the experimental results and comparative analysis, and the fourth part gives the conclusion.

2 Related Algorithms

2.1 Jaccard Algorithm

Jaccard ratio is an indicator which is used to weigh the similarity of two musters. It is defined as the intersection of two musters divided by the union of two musters. The Jaccard ratio only focuses on words with the same characteristics. The more feature words in the two sentences, the greater the value of the jacard ratio. For the two sentences S1 and S2, their Jaccard similarity is:

$$Sim(S1, S2) = \frac{|S1 \cap S2|}{|S1 \cup S2|} \tag{1}$$

The numerator is a quantity of identical terms in two sentences, and the denominator is a quantity of total terms.

2.2 Semantic Algorithm Based on Word Vector

Word2vec. Along with the popularization of deep learning in natural language processing, word vectors have also been proposed Word embedding refers to vectors that map words or phrases in the word list to the actual number through some method. The traditional One-Hot encoding converts words into discrete individual symbols, which can simply represent word vectors, but the vocabulary is often very large, which causes the vectors to be high-dimensional and sparse. Word2vec [10] can transform high-dimensional sparse vector into low-dimensional dense vector, and the position of synonyms in vector space is also close. Word2vec was released by Google in 2013 and can be used to generate and calculate word vectors. It can be effectively trained on the data set, and the training word vector can well weigh the similarity between words.

At the back of word2vec is a superficial neural network, which includes two models, one is CBOW model, and the other is Skip-gram model. Both models include input layer, hidden layer and output layer, as shown in Fig. 1. But CBOW forecasts the present word according to the context, while Skip-gram forecasts the context according to the present word. This paper is on account of the CBOW model to train word vectors. Input layer: the input is the context one hot encoding vector of the word of the selected window size; hidden layer: simple summation and average of the word vectors of the context words; output layer: this layer corresponds to a Huffman tree. The leaf nodes of the tree are the words that appear in the corpus. The non-leaf nodes are generally a two-classifier, along the left subtree is the negative class, and the right subtree is the positive class, thus calculating the probability of each word. For the sake of simplifying the complexity of the model, in comparision with the neural

probabilistic language model, the CBOW model changes the stitching method to the cumulative summation method in the hidden layer, and changes the linear structure of the output layer to a tree structure.

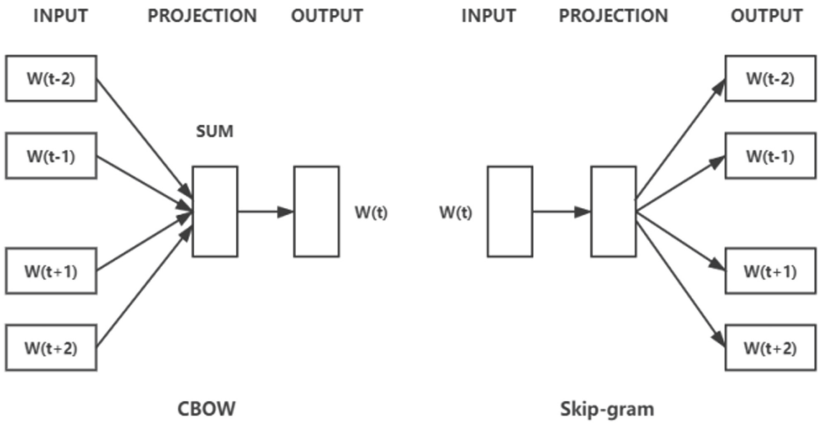

Fig. 1. word2vec model

Semantic Algorithm. The traditional Jaccard algorithm does not involve the similarity calculation at the semantic level, while the word vector can calculate the similarity of synonyms, and the sentences S1 and S2 are used to illustrate the specific algorithm steps:

Step 1: Use word2vec to train the corpus to generate model and word vectors of each vocabulary;

Step 2: Segment the sentences S1 and S2 and remove the stop words. The words in S1 are ai (i = 1,2,3, ..., m), and the words in S2 are bj (j = 1,2,3, i, n);

Step 3: Compute the semantic similarity of each word in S1 and each word in S2 through the word vector to form a two-dimensional matrix M. The formula is:

$$cos(ai, bj) = \frac{ai \cdot bj}{|ai| \times |bj|} \tag{2}$$

Step 4: Find the largest value in the matrix M, add it to the set P, and assign the value of the row and column corresponding to the value to -1, repeat this step until all the values in the matrix are -1;

Step 5: Add all the values in the set P and divide by the average similarity obtained by the set length n, as the final semantic similarity of the two sentences. The formula is:

$$Sim(S1, S2) = \frac{\sum_{i=0}^{n-1} P(i)}{n} \tag{3}$$

2.3 Algorithm Based on Jaccard and Semantics

Calculate the Jaccard similarity and semantic similarity for sentences S1 and S2, and record them as Sim1 (S1, S2) and Sim2 (S1, S2), and mix the two with a certain weight to get the final similarity. The formula is:

$$Sim(S1, S2) = \alpha \cdot Sim1(S1, S2) + (1 - \alpha) \cdot Sim2(S1, S2) \tag{4}$$

Among them, α is the weight adjustment factor, and the specific value is analyzed in the experiment. The obtained sentence similarity Sim (S1, S2) needs to be compared with the similarity threshold β, if it is greater, it is determined to be similar, otherwise it is not similar.

3 Experiment Design and Result Analysis

3.1 Experimental Details

In this article, three data sets are used to verify the algorithm. Each data set has human annotations. Two sentence labels with the same semantics are 1, and the other is 0. Data set I is a MSRP data set [11] which is provided by the Microsoft Research Interpretation Corpus. It has many positive categories and comes from news sources on the Web. Data set II is the STS data set [12], which can be used to measure the similarity of the meaning of sentences and has more negative categories than positive categories. Data set III selects 2000 data from Quora data set, and the positive-negative ratio is 1:1.

In order to reduce experimental errors and improve accuracy, all text is pre-processed, uppercase is converted to lowercase, and useless information such as punctuation marks and extra spaces is removed. Use the tf-idf method to extract keywords in the text during testing. After the above processing, all sentences are segmented and trained with word2vec to generate the model and the word vector corresponding to the word. The CBOW model is used during training, and the vector is 200-dimensional. The training corpus consists of two parts, one is the English Wikipedia corpus, the size is about 500M, and the other is the data used. This is to prevent the data of the data set from being included in the word vector model and unable to calculate the semantic similarity. The semantic similarity between words and expressions can be computed by loading the model.

The evaluation standard uses the precision rate P, recall rate R and F score commonly used in the field of information retrieval to estimate the performance of the algorithm. Precision rate P = correct prediction is positive/all predictions are positive, recall rate R = correct prediction is positive/all actual is positive, precision and recall rate affect each other, ideally both are high, but the general case if the precision rate is high, the recall rate is low, and the recall rate is low

while the precision rate is high. Therefore, in the case where both require high, the F value can be used to measure. The definition of F is:

$$F = \frac{2 \times P \times R}{P + R} \tag{5}$$

This article includes two experiments. One experiment is how to ascertain the similarity threshold of two sentences and the respective weights of the two algorithms. Another experiment is to prove the validity of the short text similarity algorithm put forward in this article, using the weighting factors obtained in the first experiment to mix Jaccard and semantic algorithms, and compare with other algorithms.

3.2 Experimental Results and Analysis

Value of Similarity Threshold β and Weighting Factor α: To determine whether two sentences are similar, a similarity threshold must be set. A smaller similarity threshold will determine that two dissimilar sentences are similar, and a larger similarity threshold will also cause a judgment error, so it is necessary to select a suitable similarity Degree threshold. The specific method is as follows:

1) Divide the interval according to the similarity value. Set an integer m and divide the interval [0, 1] into[0, 1/m), [1/m, 2/m)...[(m −1)/m,1];
2) Select the minimum and maximum values of each interval and generate a series of values evenly distributed between them;
3) Find the threshold with the lowest error rate and its corresponding accuracy rate in each interval, and record it;
4) Screen the threshold, the array Z1 and Z2 record the threshold and accuracy after screening respectively;
5) The normalized accuracy is weighted and summed with the similarity threshold to obtain the final similarity threshold

In the experiment, the data set I and the data set II are combined to find the similarity threshold and weight factor, m is set to 10, and finally the similarity is 0.58. The weighting factor is calculated according to the determined similarity threshold, and take different values of α in 0.05 steps. The result is shown in Fig. 2:

It can be found from Fig. 2 that with the α increases, the precision rate gradually increases, while the recall rate gradually decreases, and the F value increases first and then decreases. Taking the F value as the selection criterion, when α is 0.35, the maximum F value is 0.734, at this time the precision rate is 0.651, and the recall rate is 0.842. Therefore, this paper takes a weighting factor of 0.35 for subsequent performance evaluation.

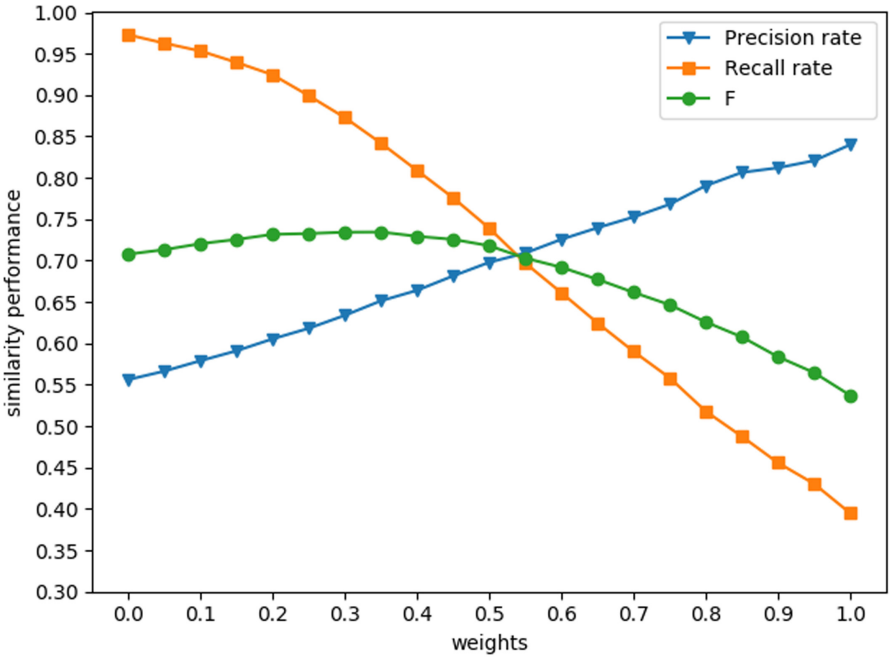

Fig. 2. Comparison of experimental results with different weighting factors

Similarity Algorithm Performance Evaluation. For the sake of proving the performance of the algorithm put forward in this article, the algorithm of this paper is now compared with several classic algorithms. The experiments are conducted in Data Set I, Data Set II and Data Set III. The classic algorithms are as follows:

Method 1: Traditional Jaccard algorithm;
Method 2: Vector-based cosine similarity algorithm;
Method 3: Edit distance algorithm based on terms;
Method 4: This article is based on a hybrid algorithm of Jaccard and semantics.

The results of the experiment are shown in the following table:

Table 1. Data set-I evaluation results.

	Method1	Method2	Method3	Method4
Precision rate	**0.892**	0.781	0.878	0.756
Recall rate	0.428	0.763	0.427	**0.867**
F	0.579	0.772	0.575	**0.807**

Table 2. Data set-II evaluation results.

	Method1	Method2	Method3	Method4
Precision rate	**0.764**	0.605	0.653	0.542
Recall rate	0.349	0.682	0.437	**0.810**
F	0.479	0.641	0.524	**0.649**

Table 3. Data set-III evaluation results.

	Method1	Method2	Method3	Method4
Precision rate	**0.667**	0.651	0.639	0.635
Recall rate	0.334	0.641	0.389	**0.793**
F	0.445	0.646	0.484	**0.705**

Table 1, Table 2 and Table 3 show the text similarity calculation performance of different methods on dataset I , dataset II and dataset III. From the experimental results, no matter on which data set, compared with the other three algorithms, the algorithm put forward in this article that combines jaccard and semantics has a significant improvement in recall rate and F value, but the precision rate Lower. The traditional jaccard algorithm only pays attention to the word form, without considering the semantics of the term, and its recall and F value are low. The vector-based cosine similarity algorithm which concentrates on the word form and its number is more stable, the accuracy and recall are not much different, and the F value is also higher. The word-based edit distance algorithm considers the word order to a certain extent, but also does not consider the semantics of the sentence. The evaluation results are similar to the jaccard algorithm. The algorithm in this paper not only considers the item information of co-occurrence terms, but also considers the semantic information of non-co-occurrence terms, and obtains good results. Comparing the evaluation results of the three data sets, the algorithm proposed in this paper, in terms of F value, data set I > data set III > data set II, which may be related to the characteristics of the data set. Data set I has more positive classes than negative classes, and data set II has more negative classes than positive classes, and the positive class is equal to the negative class in Data Set III. And the data set III has not been used to find the similarity threshold and weighting factor, but when the weighting factor is 0.35, good experimental results are also obtained.

4 Conclusion

This article puts forward a text similarity algorithm based on a mixture of Jaccard and semantics. This algorithm first considers the effect of co-occurrence words on text similarity, and uses the traditional Jaccard algorithm to compute the similarity of two sentences. Secondly, the word vectors are acquired by training the external corpus, then the corresponding values between the word vectors

in the two sentences are calculated, the maximum value is taken out and the word vectors in the corresponding two sentences are deleted, then average all the maximum values as the semantic similarity of these two short sentences. Finally, the weighted Jaccard similarity and semantic similarity are combined to compute the final similarity of the two sentences. In this paper, experiments were carried out on three data sets, and the algorithm was compared with the conventional Jaccard algorithm, cosine similarity algorithm, editing distance algorithm, etc. The results show that the algorithm of this paper is higher than other methods in the recall rate R and F of the text similarity calculation, thus proving the effectiveness of the algorithm. However, the effect of this algorithm in text similarity calculation is not very significant, which is related to the linguistic features such as word vectors obtained by training, sentence syntax, and word order. Because the larger the training corpus, the better the word vectors obtained by training, but the training corpus in this article is only a medium size, and the algorithm does not consider the semantic impact of the order of words in the sentence and the composition of the sentence on the sentence.

References

1. Erjing, C., Enbo, J.: A review of text similarity calculation methods. Data Anal. Knowl. Discov. **1**(6), 1–11 (2017)
2. Hanru, W., Yangsen, Z.: A review of research progress in text similarity calculation. J. Beijing Inf. Sci. Technol. Univ. (Nat. Sci. Edn.) **34**(01), 68–74 (2019)
3. Xianying, H., Yingtao, L., Qinfei, R.: An English short text similarity algorithm based on common chunks. J. Chongqing Univ. Technol. (Nat. Sci.) **29**(08), 88–93 (2015)
4. Zhixiang, G., Xie Longen, D.Y.: Implementation and improvement of SimHash algorithm for text similarity calculation. Inf. Commun. **01**, 27–29 (2020)
5. Li, L., Zhu, A., Su, T.: Research and implementation of an improved text similarity algorithm based on vector space. Comput. Appl. Softw. (02), 282–284 (2012)
6. Yuan, X.: Research on text similarity based on HowNet. J. Chengdu Univ. (Nat. Sci. Edn.) **33**(3), 251–253 (2014). https://doi.org/10.3969/j.issn.1004-5422.2014.03.015
7. Wanxiang, C., Ting, L., Bing, Q., et al.: Chinese similar sentence retrieval based on improved edit distance. High-Tech Commun. **14**(7),15–19 (2004). https://doi.org/10.3321/j.issn:1002-0470.2004.07.004
8. Jinpeng, Z.: Research and application of text similarity algorithm based on semantics. Chongqing University of Technology (2014)
9. Zhifang, L., Guoen, Z., Junfeng, L., Fei, L., Fei, C.: Chinese short text grammar semantic similarity algorithm. J. Hunan Univ. (Nat. Sci. Edn.) **43**(02), 135–140 (2016)
10. Mikolov, T., Chen, K., Corrado, G., et al.: Efficient estimation of word representations in vector space (2013). arXiv preprint arXiv:1301.3781
11. Dolan, W., Quirk, C., Brockett, C., et al.: Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources (2004)
12. Cer, D.M., Diab, M.T., Agirre, E., et al.: SemEval-2017 Task 1: semantic textual similarity multilingual and crosslingual focused evaluation. In: Meeting of the Association for Computational Linguistics, pp. 1–14 (2017)