International Journal of Cognitive Informatics and Natural Intelligence
Volume 15 • Issue 4

API Recommendation Based on WII-WMD

Wanzhi Wen, Nantong University, China*
Shiqiang Wang, Nantong University, China
Bingqing Ye, Nantong University, China
XingYu Zhu, Nantong University, China
Yitao Hu, Nantong University, China
Xiaohong Lu, Nantong University, China
Bin Zhang, Nantong University, China

ABSTRACT

Improving software development efficiency based on existing APIs is one of the hot researches in software engineering. Understanding and learning so many APIs in large software libraries is not easy, and software developers prefer to provide only requirements descriptions to get the right API. In order to solve this problem, this paper proposes an API recommendation method based on WII-WMD, an improved similarity calculation algorithm. This method firstly structures the text and then fully mines the semantic information in the text. Finally, it calculates the similarity between the user's query problem and the information described in the API document. The experiment results show that the API recommendation based on WII-WMD can improve the efficiency of the API recommendation system.

KEYWORDS

API Recommendation, Code Recommendation, Machine Learning, Natural Language Processing, Similarity Calculation, Word Embedding

1. INTRODUCTION

Nowadays, with the rapid development of information technology, various improved algorithms are proposed to improve the technologies' effectiveness (Li et al. 2018, Li et al. 2019a, Li et al. 2019b, Wang et al. 2018, Wang et al. 2019). Furthermore, software products are becoming more and more complex, and APIs in software libraries are becoming more and more abundant. Improving software development efficiency based on existing APIs has become one of the hot researches in software engineering. However, understanding and learning so many APIs in large software libraries is not easy (Ko et al. 2004), and software developers prefer to provide only requirements descriptions to get the right API. Existing keyword retrieval methods are difficult to identify lexical and syntactic differences between requirement description and API documents, which leads to low efficiency of API recommendation.

In order to improve the recommendation efficiency, researchers proposed many API recommendation techniques (Mcmillan et al. 2011, Chan et al. 2012, Holmes et al. 2005, Rahman et al. 2017, Goldberg et al. 2014, Hoecker et al. 1995, Eggert et al. 2004, Bengio et al. 2009, Jang et al. 2016, Ma et al. 2015), which include recommendation methods based on semantic and no-semantic

DOI: 10.4018/IJCINI.20211001.oa16 *Corresponding Author

information. Word embedding based API recommendation is one of the most popular techniques. Word embedding is a way to transform words in text into the form of vector representation (Lai et al. 2016). The simplest method of word embedding is one-hot coding based on the word bag model (Karakasis et al. 2015). One-hot coding is the most basic vector representation method, in which N bit state register is used to encode N states, and the text vector is used to represent the words in the text, where only the position corresponding to the word is 1 and all other positions are 0. The commonly-used technique is Word2Vec (Frome et al. 2013), which is based on the CBOW model and Skip-gram model. Both models are based on the three-layer structure in the neural network language model. In this method, each word in the text is represented by vector and the word vector with similar semantic information has close spatial position. The position of synonyms in vector space will be closer, which guarantees semantic information in the text (Lilleberg et al. 2015).

In general, there are several ways a developer can choose to query an API. The first one is search engine. Studies show that 70% of developers often use search engines to find the appropriate API Search engines like Baibu and Google can get some API information (Brandt et al. 2009), which use specific strategies to retrieve valuable information from the Internet and feed it back to developers (Ko et al. 2006). The second one is API help document. Once developers have determined which API you are using, you can find its usage, function description, code specification, and so on directly in the corresponding API help document. The third one is developer forum such as Stack Overflow. Developer forum is a technical Q&A site that is currently one of the most popular developer communities. This site provides a platform for developers to exchange experiences with APIs. Users can submit questions and view relevant content posts for free on the site. However, these methods also have different problems, Search engines search for regular web pages, rather than APIs designed specifically for development, such as source code and function parameters, which are program-related. API help document and developer forum need developers' experiment (Duala-Ekoko et al. 2012).

Based on above, we proposed an API recommendation method based on WII-WMD to improve the recommendation efficiency. The API recommendation method based on WII-WMD improves a text similarity calculation based on deep mining semantic information through word embedding technique. Firstly, the paper proposes an improved method WV-IP-IDF, which introduces the part-of-speech influence factor and the information entropy factor based on TF-IDF. Then we combined WV-IP-IDF algorithm with WMD (Word Mover 's Distance) and put forward new method of similarity calculation WII – WMD. Finally, the experiments verified the effectiveness of the algorithm.

To assess WII-WMD algorithm based API recommended method, we extracted Q&A from Java Stack Overflow and verified the effectiveness of the algorithm by experiments. According to the experimental results, the API recommendation method based on WII-WMD is about 1% higher than that based on WV-TF-IDF in MRR and MAP indexes respectively. In the WII-WMD similarity algorithm, the IDF original algorithm introduces the influence factor of part-of-speech and information entropy and the search performance of the system is improved. The VSM-TF-IDF method only uses the space vector model, does not consider the semantic information in words, and only relies on the matching of keywords to complete the calculation, so overall performance of VSM-TF-IDF method is poor.

The main contributions of this paper are:

1. An improved algorithm WV-IP-IDF based on TF-IDF is proposed. Firstly, we preprocessed the text by word embedded modeling, and then TF-IDF algorithm was used to weight the text information represented by word vectors.
2. Combining WV-IP-IDF and WMD, we proposed WII-WMD, an improved similarity calculation algorithm WII-WMD. WMD word vector doesn't consider feature extraction and only uses normalized word frequency information. The combination can improve the accuracy of similarity calculation by reducing noise data input.

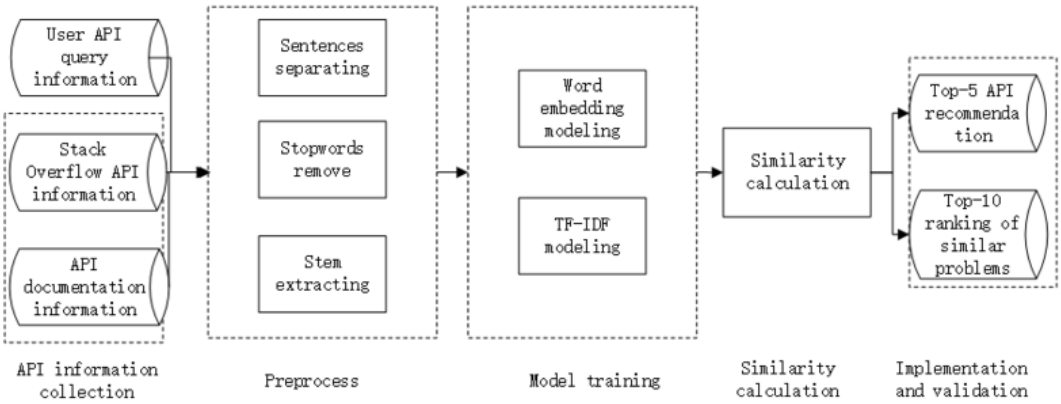
3. Based on the above researches, API recommendation based on WII-WMD is proposed. The API recommendation system based on word embedding technique proposed in the paper is implemented, and the recommendation capability of the system is verified through case analysis and experimental research. The validity of the system is evaluated by MRR and MAP.

The remainder of the paper is organized as follows. Section II describes our approach framework. Section III describes the empirical study, including the datasets and the performance measure. Finally, Section IV discusses the conclusion and our future work.

2. OUR APPROACH

2.1. Overall Framework

Figure 1. Overall framework



The overall framework is shown in Figure 1. The detail steps are as follows.

- API information collection. The content of the API help document and API information, including methods in the API, description of the API, the class of the API and other information, is collected. Furthermore, the test set and API document database of API information are segmented and reconstructed. XML parsing of Java tag data in Stack Overflow and Q&A pairs including questions, answers, question ratings, answer ratings and tag information are also stored in the database.
- Preprocess. We preprocessed the collected data, including sentences separating, stop words removing, stem extracting and part of speech tagging.
- Model training. Word embedding modeling and TF-IDF modeling are carried out for the preprocessed text information. We converted all words into corresponding word vectors, generated word models and saved the value of TF-IDF.
- Similarity calculation. Combining WV-IP-IDF and WMD, we proposed WII-WMD, an improved similarity calculation algorithm WII-WMD.
- Implementation and validation. The API recommendation system based on word embedding technique is implemented, and the recommendation capability of the system is verified through case analysis and experimental research. We used the improved similarity computing algorithm WII-WMD to calculate the similarity between the user problem and the problem data set. By

calculation, the top-10 problem sets are sorted by similarity. The top 10 APIs in the database are obtained from these top 10 candidate questions. Based on the harmonic mean of the document similarity, the candidate top 5 API is finally recommended to the developer.

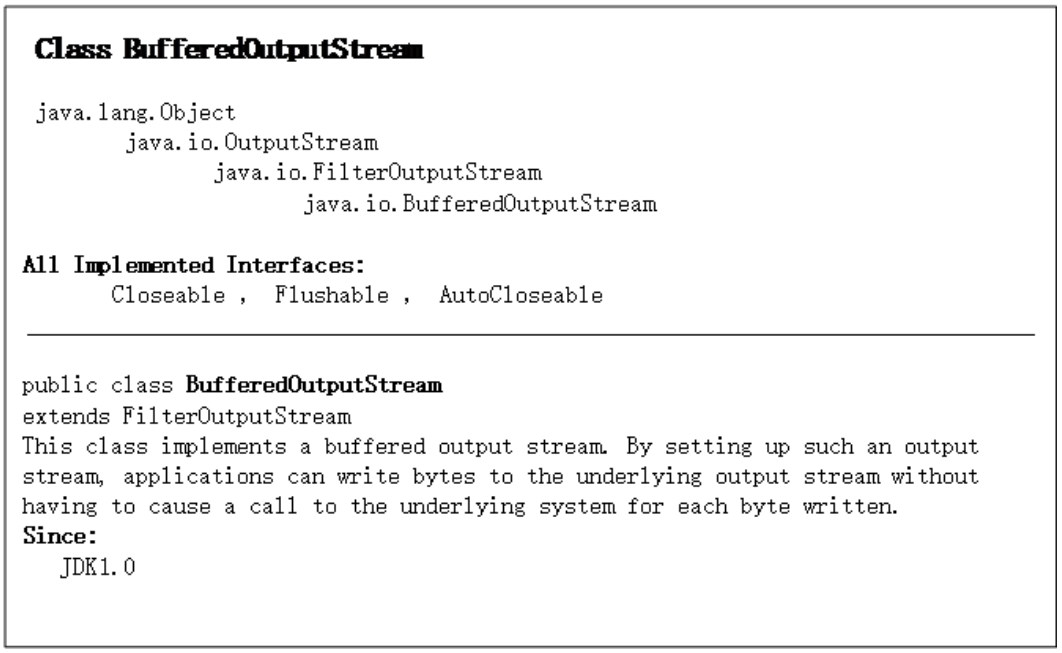
2.2. Data Collection

The API help document and the Q&A on the Stack Overflow site contain a lot of API information.

2.2.1. API documentation

API documentation contains detailed API information and is the most direct and effective way to learn APIs. The API help document usually introduces the inheritance relationship of API and the usage description of function. For different interfaces and abstract classes, and it also systematically introduces the corresponding constructor and the built-in function. In addition, some API help documentations also contain the examples of code use cases. An example of Java help document is shown in Figure 2.

Figure 2. An example of Java help document



API documents are usually stored in HTML format on websites for developers to search and learn online. The crawler tool makes it easy to analyze the contents of the help document. In our experiments, we obtained the method name MethodName, the class name BelongsClass, and API description Description and we stored the extracted contents in the database

2.3. Q&A on Stack Overflow

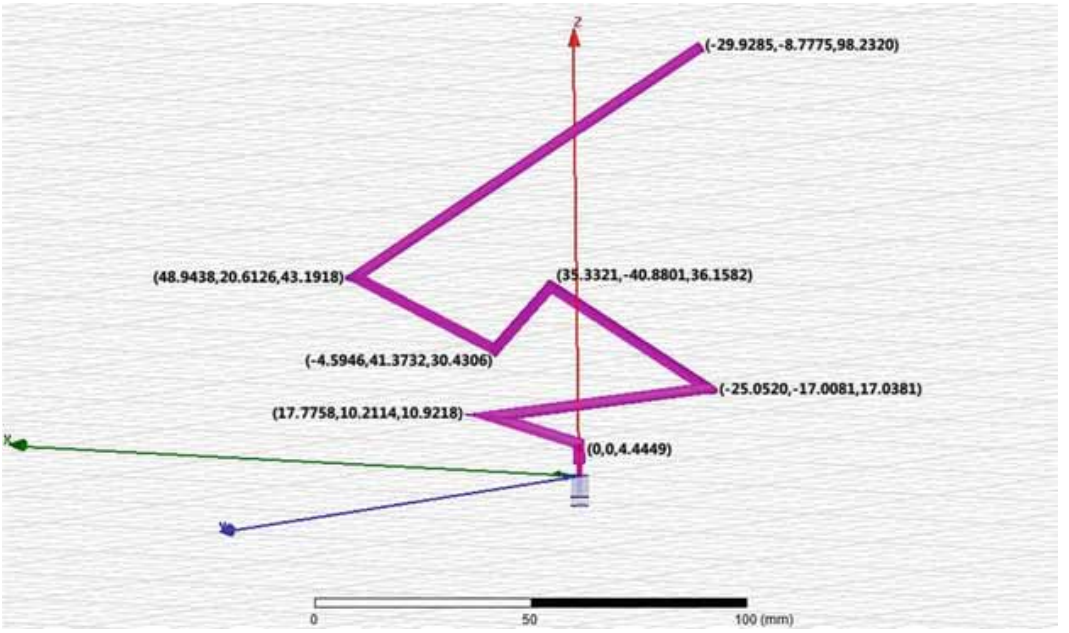
Stack Overflow is a technical Q&A website, which is one of the most popular developer communities. Users can submit questions on the site for free and browse questions to find relevant content. Stack Overflow contains not only various answers and code samples for the questioner, but also other

information such as user ratings, user adoption, tag information, the best answers, and information about the questioner and respondent. An example of Q&A information in Stack Overflow is shown in Figure 3.

In our experiment, we extracted API recommendation information of Java projects and the extracted Q&A pairs contain Java tags. The threshold of questions' scores is 5 points. The answer of the question contains API entities and the score of the answer should be positive. So the validity of getting data from Stack Overflow is guaranteed. Generally, posts that contain answers and have high rates for their answers are of higher value.

Therefore, this experiment extracts the keywords from data dump published by Stack Overflow as the data set. The data in data dump has been saved in XML form.

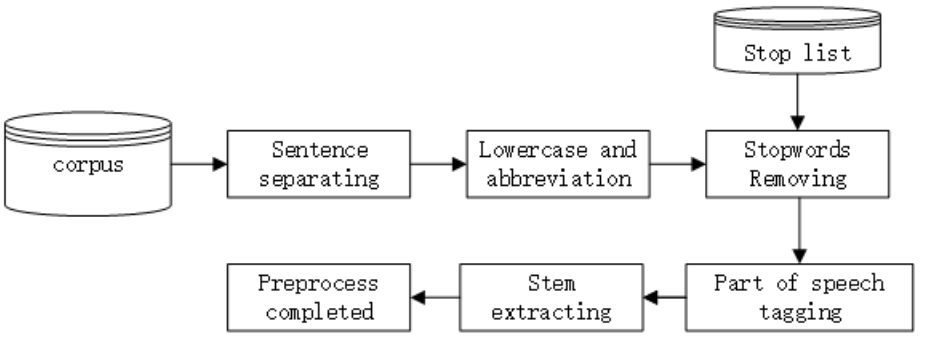
Figure 3. An example of Q&A information in Stack Overflow



3. PREPROCESSING

No matter the user's API query problem or the description in the API document, data preprocessing is required before further calculation. Preprocess has two functions: one is to eliminate the influence of interference data, reduce data noise, and make the data neat; the other is to improve the efficiency of model calculation based on the preprocessed data. The implementation in the preprocessing process is based on the NLTK library in Python. The pre-processing process is shown in Figure 4.

Figure 4. Preprocessing



Sentence separating. The corpus of the API recommendation is from API help document and Q&A pairs of Stack Overflow. The structure of sentences is characterized by the separation of period, question mark and exclamation mark. We separated the sentences by using the Sentence_Tokenizer method in Tokenizer package.

Word segmentation and lowercase processing. The WordPunctTokenizer method in the Tokenizer package is also used for word segmentation. Words are separated by Spaces. In the case of hyphenated words, such as "peace-loving", we should remove all hyphens . At the same time, we converted all the words from uppercase to lowercase .

Abbreviate completing. Abbreviations are commonly used in English, such as " 's "," re ", " 'm ". We listed all the abbreviate and use replace method to complete abbreviate, such as Replace (' \ 'd ', 'would').

Stopwords Removing. NLTK library has a Stopwords table, which contains words without actual meaning, such as "a", "with", "and", etc. We removed all the stopwords.

Part of speech tagging. In this paper, the weight information of the target word is changed through the influence factor of part of speech. We used the pos_tag() method to implement part of speech tagging of words. The list of parts of speech obtained by pos_tag() method is shown in Table 1.

Table 1. Commonly used parts of speech

parameter	Part of speech	parameter	Part of speech
NN	nounplural	RB	abverb
NNP	propernoun	JJ	adjective
VB	verb	WP	wh-pronoun

Stem extracting. This step restores "ed" and "ing" in the tense and passive voice, "s" and "es" in the plural and third person singular, and reduces repetition. The snowballStemmer() method is used to extract similar stems.

Algorithm 1. shows the details of the preprocessing steps

Algorithm 1 Preprocessing algorithm
Input: Corpus **Output: Corpus data1 and part-of-speech tagging data set data2 completed by preprocessing**
1. for text in corpus: **2.** sentecnes = text.Sentence_Tokenizer **3.** for sentence in sentences: **4.** List_word = WordPunctTokenizer(sentence).lower() **5.** for i in List_word **6.** if i not in stopwords.words('English') **7.** List_OutStopWords ¬ i **8.** end if **9.** end for **10.** end for **11.** end for **12.** data1 = [ ],data2 = [ ] **13.** for i in List_OutStopWords: **14.** data1 .append(i. snowballStemmer()) **15.** data2 .append(i. pos_tag()) **16.** return data1,data2

4. MODEL TRAINING

In order to calculate the similarity between the developers questions and the descriptions in Stack Overflow or the API help document, we processed the API model training.

The previous section completed the preprocessing of the information in the knowledge base before word embedding modeling. The data has been formatted into the input style required in word embedding model. Based on this preprocess, the corpus is simplified, which improves the efficiency of modeling. Modeling includes two stages, Word embedded modeling and TF-IDF modeling.

Word embedded modeling. Word2Vec is the most commonly used word embedding modeling technique. When the model is initialized, Word2Vec computes the data through a three-layer structure, namely the input layer, the output layer, and the hidden layer.

This paper uses the Gensim package in python to invoke the Word2Vec for modeling. The definitions of parameter values during model training are shown in Table 2.

Table 2. Parameter settings during model training

Parameter names	Parameter Description	Parameter value
embedding_size	The vector dimension	100
skip_window	Sliding window size	5
num_skips	Sliding value	2
num_steps	Steps	100000
num_sampled	Sample value	64
vocab_size	Glossary size	50000
learning_rate	Learning rate	0.0001
epoch	Number of training cycles	100
batch_size	Number of samples in batch	100

TF-IDF modeling. We used the NLTK package in python to complete this modeling process. During the modeling, TextCollection method is used to build the corpus, and then tf() and idf() methods are used to get the TF and IDF values of the words in the corpus.

5. SIMILARITY CALCULATION BASED ON WII-WMD

We firstly combined the weighted TF-IDF and the word vectors to get the new word vectors, and then introduced the influence factor of part of speech and information entropy to propose WV-TF-IDF algorithm. Based on WV-TF-IDF, similarity calculation based on WII-WMD is proposed.

Word2Vec is used for word embedding modeling to generate word vectors containing semantic information. The API keywords are extracted by TF-IDF weighted algorithm. The word vectors are obtained from training text by Word2Vec. Combining the weighted TF-IDF and the word vectors, we got the new vectors as Formula 1.

$$\mathrm{vecC} = \sum\nolimits_{k \in C} v_k \cdot tfidf\left(k, C\right) / h \tag{1}$$

In Formula 1, v_k denotes the word vector of Keyword k, $tfidf(k, C)$ denotes the weight of Keyword k in API Information C, h is the length of API Information C.

Then we proposed WV-TF-IDF algorithm by introducing the influence factor of part of speech and information entropy. The detail of WV-TF-IDF algorithm is shown in Algorithm 2.

In the API recommendation, the similarity between the developer's requirements and the description in the API document is calculated. The text or sentence is taken as a whole, and the overall similarity is calculated according to each word. The keywords in the text or sentence should take up a larger proportion in similarity calculation.

The requirements and descriptions in the API recommendation are converted to weighted vectors by WV-IP-IDF algorithm as the input vectors of WMD and we proposed WII-WMD based on this improvement.

Algorithm 2 WV-TF-IDF algorithm

Input: data set M (containing m documents), word vector v_i trained by Word2Vec model
Output: new vector v_i''

\\calculate the inverse document frequency IDF for all words in M

for each word w_i

$idf_i = \log\left(m / m\left(w_i\right)\right)$

end for

\\calculate the part of speech influence factor $POS\left(w_i\right)$ of all words in M

for each word w_i

if w_i is a noun

$POS\left(w_i\right) = 0.57$

if w_i is a verb

$POS\left(w_i\right) = 0.24$

if w_i is an adverb or an adjective

$POS\left(w_i\right) = 0.15$

else

$POS\left(w_i\right)) = 0.04$

end for

\\calculate the information entropy factor I for all words in M

for each feature w_i

$\mathrm{I} = f\left(H\left(w_i\right)\right) = log_2 n / H\left(w_i\right)$

end for

$v_i'' = vecw_i = v_i \cdot f\left(H\left(w_i\right)\right) \cdot POS\left(w_i\right) \cdot idf\left(w_i, M\right)$

return v_i''

Let the requirement and description of API be text d and text d' respectively. The steps of similarity calculation are as follows:

Step 1: The corpus is preprocessed, including sentence separating, word segmentation and lowercase processing, abbreviate completing, stopwords removing, part of speech tagging and stem extracting, etc. The content of preprocessing is detailed in Section C.

Step 2: The data in the corpus is trained to collect the word vector based on Word2Vec. The IDF value is got by TF-IDF model, and the part of speech information POS is obtained through the part of speech analytic model.

Step 3: WV-IP-IDF algorithm is used for weighted word vector representation of text d and d' words, $\mathrm{d} = \{ \mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_m \}$ and $\mathrm{d}' = \{ \mathrm{w}_1', \mathrm{w}_2', \ldots, \mathrm{w}_n' \}$.

The WMD distance between two texts is calculated. Because Similarity is inversely proportional to distance, the similarity calculation formula between d and d' can be obtained as Formula 2.

$$Simd,d' = \frac{\partial}{\partial + WMD\left(d,d'\right)} \tag{2}$$

Where ∂ is the equilibrium factor, which usually takes the value of 1.

Algorithm 3. shows the detailed flow of the WII-WMD.

Algorithm 3 WII-WMD algorithm
Input: data set M, word vector v_i **trained by Word2Vec model** **Output: Similarity between text d and** d'
calculate the inverse document frequency IDF for all words in M **calculate the part of speech influence factor** POS of all words in M **calculate the information entropy factor** I of all words in M (see Algorithm 2 for calculation IDF□ POS□ I **)** $w_i = veci = v_i \cdot idf\left(i,M\right)POS\left(i\right)\cdot f\left(H\left(i\right)\right)$ **weighted vector between text d and** d' represents d={ w_1, w_2,…, w_m } and d' ={ w_1', w_2',…, w_n' } $Simd,d'$ **is calculated by Formula 2** **return** $Simd,d'$

6. IMPLEMENTATION AND VALIDATION

Based on WII-WMD, we implemented the API recommendation. The process of recommendation include three times of similarity calculation.

First similarity calculation. The similarity between the users' questions and questions in the Stack Overflow table in the database is calculated by the WII-WMD. The similarity is denoted as Sim_1 and the top-10 most relevant questions are obtained according to Sim_1. Then, the APIs corresponding to the answer of the top-10 most relevant questions in the data table is obtained, that is, the candidate APIs are obtained.

Second similarity calculation. After candidate APIs are obtained after the first calculation, all the corresponding API description information can be found in the API document data table. WII-WMD is used to calculate the similarity between the users' requirements and candidate APIs' descriptions again. In this paper, we denote the second similarity as Sim_2.

Third similarity calculation. The harmonic average of the two similarities is calculated as the final similarity score, denoted as Sim_3. Finally, the top-5 API information is recommended according to the similarity ranking. The third similarity is calculated as Formula 3.

$$Sim_3 = \frac{2Sim_1Sim_2}{Sim_1 + Sim_2} \tag{3}$$

7. EMPIRICAL STUDY

7.1. Research Question

In this paper, we conducted experiments to study the following four questions.

RQ1: How about the similarity calculation between sentences with our proposed WII-WMD similarity algorithm? Is it better than the four similarity calculation algorithms of VSM-TF-IDF, WV-TF-IDF, Doc2Vec and WMD algorithm?

RQ2: Can our API recommendation system help users solve query problems well?

RQ3: Does our WII-WMD similarity algorithm have a better effect on API recommendation than VSM-TF-IDF and WV-TF-IDF?

RQ4: The word embedding technique we used is based on the Word2Vec model, does it work better than other word embedding models?

7.2. Datasets and Evaluation Metrics

We use open dataset SICK and the STS to evaluate the effectiveness of similarity calculation and use Q&A information in Stack Overflow and APIs information in JAVA SE8 to evaluate the effectiveness of API recommendation.

In our experiment, we firstly calculated the similarity score based on VSM-TF-IDF, WV-TF-IDF, Doc2Vec, WMD, WII-WMD. Pearson correlation coefficient is used to evaluate the effectiveness of similarity calculation (Qin et al. 2013). The value of pearson correlation coefficient can be got from Formula 4.

$$\mathrm{p} = \frac{\sum \mathrm{XY} - \frac{\sum \mathrm{X} \sum \mathrm{Y}}{\mathrm{N}}}{\sqrt{\left(\sum \mathrm{X}^2 - \frac{(\sum \mathrm{X})^2}{\mathrm{N}}\right)\left(\sum \mathrm{Y}^2 - \frac{(\sum \mathrm{Y})^2}{\mathrm{N}}\right)}} \tag{4}$$

In the above formula, variable X represents the sentence similarity calculated by similarity calculation method, variable Y represents the similarity answer given in the data set, N represents the number of sentence pairs in the text. The greater the value of Pearson correlation coefficient means is, the better the calculation result is.

We evaluated the effectiveness of API recommendation by mean reciprocal ranking MRR and mean accuracy MAP.

MRR is a commonly used evaluation metric in search algorithm. The inverse of the ordinal number of the correct answer in the result is taken as the accuracy, and MRR is the average of all inverses. Its calculation is as Formula 5.

$$\mathrm{MRR} = \frac{1}{\left|Q\right|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{5}$$

In the above formula, $\left|\mathrm{Q}\right|$ is the number of queries and $\mathrm{rank_i}$ is the ordinal number of the correct answer to the *i*th query.

MAP is an evaluation metric used in the field of information retrieval to measure the ranking performance of search engines. It evaluates the average query accuracy.

In Formula 6, MAP is the mean of the AP's for Q query problems. AP is the average accuracy of each query.

$$\mathrm{MAP} = \frac{\sum_{q=1}^{Q} AP\left(q\right)}{Q} \tag{6}$$

8. RESULTS AND ANALYSIS

i. RQ1: This paper uses the Word2Vec word embedded technique. Based on the Skip-gram model, we set the parameters and train the word vector (Krishnamurthy et al. 2016). The parameters are set as follows: the vector dimension is 100 dimensions, the window size is 5, word vector iteration number is 50, iteration step length is 0.00005 in a stochastic gradient descent method.

After the texts are preprocessed, we calculated the similarity between sentences by WII-WMD. Then we compared the similarity score with that of VSM-TF-IDF, WV-TF-IDF, Doc2Vec and WMD. The comparison results are shown in Figure 5 and Figure 6. The abscissa is the serial number of the sentence pairs and the ordinate is the similarity score of the sentence pairs. In Figure 5 and Figure 6, the solid line is the similarity score of random 60 sentence pairs in the data set.

In Figure 5, it can be found that the three similarity calculation methods differ greatly from the similarity score provided by the data set, while the similarity score calculated by the WII-WMD and WDM algorithm are closer to the score provided by the data set in Figure 6. What's more, the WII-WMD algorithm proposed in this paper performs better than WMD algorithm.

To further evaluate the effectiveness of similarity calculation methods. Pearson coherence coefficient is used to verify the results of these five similarity calculation methods in all data sets. The experimental results are shown in Table 3.

Figure 5. S`imilarity comparison of Doc2Vec, VSM-TF-IDF and WV-TF-IDF`

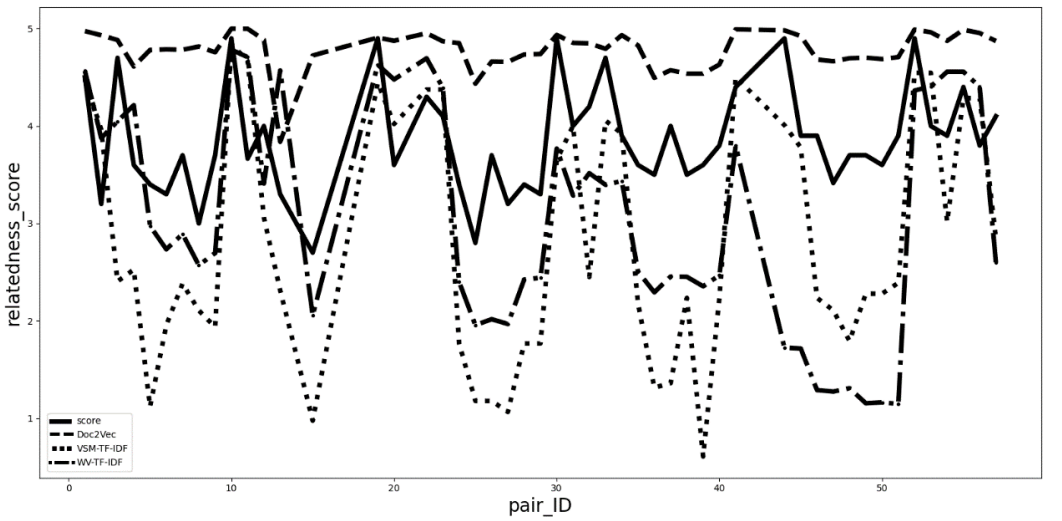


Figure 6. Similarity comparison of WMD and WII-WMD

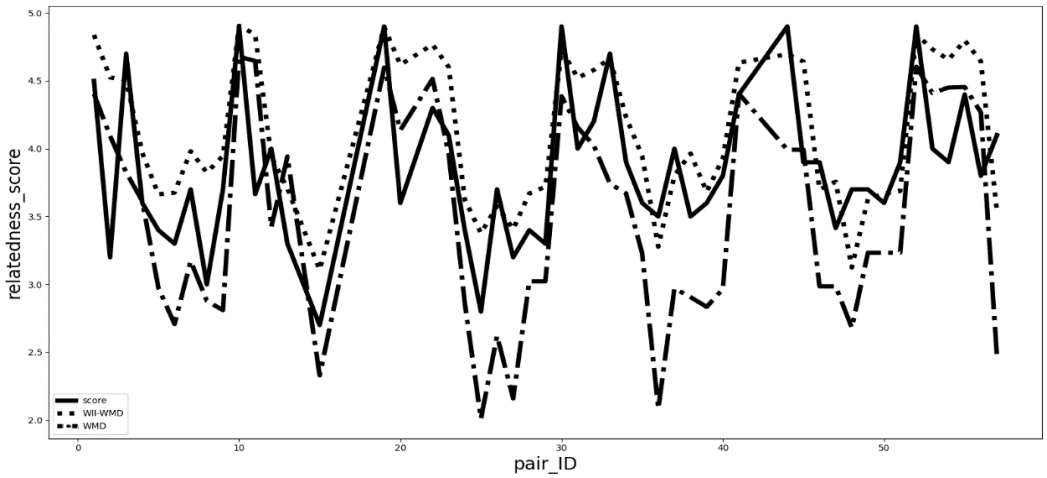


Table 3. Experimental results of five algorithms

Method	Pearson（%）	Method	Pearson（%）
VSM-TF-IDF	59.73	WMD	63.24
WV-TF-IDF	59.91	WII-WMD	69.47
Doc2Vec	41.44		

Obviously, the value of Pearson coherence coefficient of WII-WMD is greater than that of other methods, which indicates WII-WMD performs best. tWord2Vec and Doc2Vec methods are based on the Skip-gram model. During sentence training process, Doc2Vec introduces the word order characteristics in the text and the value of similarity is calculated similarly to the WV-TF-IDF. Therefore, the difference is small between Doc2Vec method and WV-TF-IDF method .

Compared with other algorithm, WII-WMD improved WV-TF-IDF most. The spatial vector model used by VSM-TF-IDF algorithm ignores the semantic relation between words. The other four methods all use Word2Vec model for language modeling, and the word vector contains semantic information, which results in the difference between similarity calculation. Compared with WMD algorithm, WII-WMD improve the value of Pearson correlation coefficient. The reason is WII-WMD algorithm takes into account the influence of part of speech and formation entropy and makes full use of text information when measuring text similarity.

ii. RQ2: We implemented Java API recommendation in JAVA, collect more than 100,000 question and answer pairs with Java label in Stack Overflow, crawl API information in Java SE8 help documents, and finally store them in the database. To verify the effectiveness of our API recommendation based WII-WMD, this section randomly selects five Java programming issues from the test set, as shown in Table 4. The first two recommendation result of our method is shown in Table 5.

Table 4. Five programming problems extracted from Stack Overflow

Rank	Extract problem items
1	How to initialize each element of an array with a specific value efficiently?
2	How to generate a random combination of Numbers in Java?
3	How to round a number to n decimal places in Java?
4	How to initialize a TreeMap with pre-sorted data?
5	How to force terminate all workers in ThreadPoolExecutor immediately?

Table 5. Two API recommendations for five programming problems

Rank	API recommendation result 1	API recommendation result 2
1	java.util.Arrays.fill	java.lang.System.arraycopy
2	java.util.Collections.shuffle	java.lang.Math.random
3	java.lang.Math.round	java.lang.Double.toString
4	java.io.BufferedReader.readLine	java.nio.file.Files.lines
5	java.lang.Runtime.exec	java.lang.System.exit

Take the first question as an example. The first question is " How to initialize each element of an array with a specific value efficiently". First, we preprocessed the question. The result is "'how', 'initialis', 'elemen', 'array', 'specifi',, 'java', 'constant', 'valu', 'effici'", and then the word embedding model and TF-IDF model are used to calculate the similarity. The first calculation is to calculate the similarity between the input questions and the questions in Stack Overflow, and get the top-10 candidate questions. The contents of the 10 candidate questions are shown in Table 6. After the candidate problem set is obtained, the API entity information corresponding to the problem can be obtained. Then, we combined the similarity of the problem and API description to obtain the final similarity score and top-5 API recommendations. Finally, we completed the recommendation of all the content.

The recommended top-5 API content and similarity value are shown in Table 7. The first recommendation API is java.util.arrays.fill, which is very close to the initial problem " How to initialize each element of an array with a specific value efficiently ". In the Java help document, the Arrays class in Java.util contains various methods for manipulating Arrays, such as sorting and search. The fill () method in this class assigns the specified data type to an array of the specified data type. Obviously, this is an initial allocation problem with a data type, related to the user input problem. The description of the function in the recommended result also confirms the results "Assigns the specified long value to each element of the specified array". From list of top-5 questions, the questions focused

on the data types that initialized the array and these recommendations will help the user solve the problem. Through this example analysis, the API recommendation based on WII-WMD has a good effect.

iii. RQ3: We compared the effectiveness of API recommendation based on different methods, such as VSM-TF-ID and WV-TF-IDF and WII-WMD. To effectively evaluate different method, the collected data structure is shown in Table 8, including extracted high-scoring questions and corresponding answer APIs. The influence of different similarity algorithms on the recommendation system is evaluated by MRR and MAP.

Table 6. Top-10 similarity problem and similarity of query problem

Rank	Similarity problem	Similarity
1	why arrays are initialized to default values but not arraylist?	0.812
2	Array constants initialization in Java	0.762
3	initializing a boolean array in java	0.748
4	how to initialize a static array of objects in java	0.741
5	initialize the elements array to any specific value	0.733
6	How to declare an ArrayList with values?	0.726
7	How do I declare and initialize an array in Java?	0.713
8	How to initialize a dynamic array in java?	0.706
9	How to name an array with a variable in java?	0.689
10	How to initialize a list with part of an array in Java	0.664

Table 7. Top-5 API and similarity of query problems

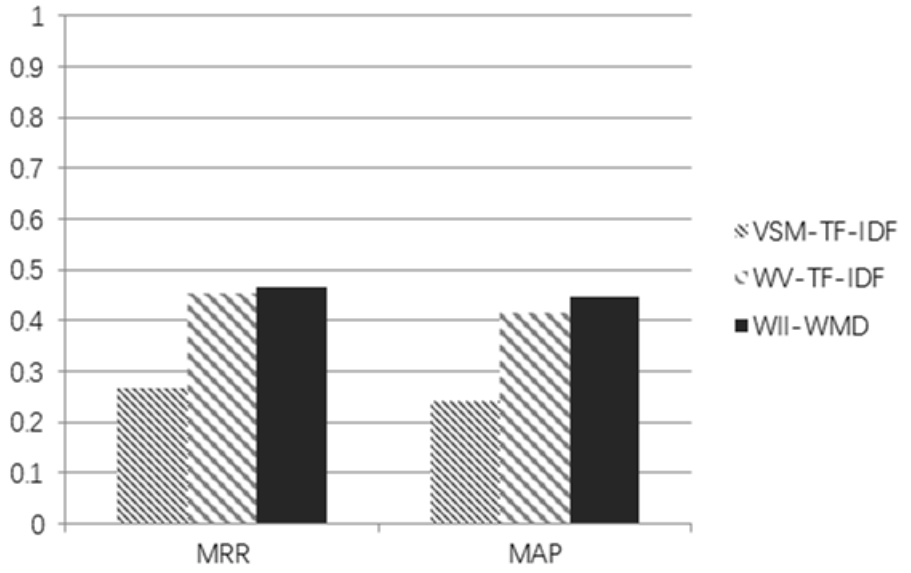
Rank	Top-5 API	Similarity
1	java.util.Arrays.fill	0.842
2	java.lang.System.arraycopy	0.767
3	java.util.Arrays.asList	0.682
4	java.util.Collections.nCopies	0.658
5	java.util.Collections.unmodifiableList	0.613

The recommended results of the three similarity calculation methods are shown in Figure 7. It can be found that the overall performance of VSM-TF-IDF method is poor, because the VSM-TF-IDF method only uses the space vector model, which does not consider the semantic information in words and only relies on the matching of keywords to complete the calculation. The WII-WMD proposed in this paper improves WV-TF-IDF by about 0.8% in MRR and 1.4% in MAP respectively. Both algorithms use Word2Vec for modeling. WII-WMD introduces the influence factor of part of speech and information entropy, the search performance of the system is improved.

Table 8. Collected partial data in the test set

query_ID	query	answer_API
1	How to collect DoubleStream to List	java.lang.Double.parseDouble
2	JAVA: Concurrency control for access to list in java	java.util.Collections.synchronizedList
3	How do I use swing in a shutdown hook?	java.lang.System.exit
4	How to ensure a piece of code is run before exiting a java application	java.lang.Runtime.addShutdownHook
5	How to timeout a thread	java.lang.Thread.interrupted
6	Splitting a string in java on more than one symbol	java.lang.String.split
7	How to hide cursor in a Swing application?	java.awt.Toolkit.createCustomCursor
8	How to generate a random permutation in Java?	java.util.Collections.shuffle
9	Java generating Strings with placeholders	java.lang.String.format
10	How to see if an object is an array without using reflection?	java.util.Arrays.toString
11	How to blur a portion of an image with JAVA	java.awt.image.BufferedImage.getSubimage
12	How to instantiate inner class with reflection in Java?	java.lang.Class.getDeclaredConstructor
13	First char to upper case	java.lang.Character.toUpperCase
14	Setting log level for inner class (in Glassfish)	java.util.logging.Logger.getLogger
15	Escaping special characters in Java Regular Expressions	java.util.regex.Pattern.quote
16	Java8 java.util.Date conversion to java.time.ZonedDateTime	java.time.ZonedDateTime.ofInstant
17	Setting the default Java character encoding?	java.nio.charset.Charset.defaultCharset
18	Is there any way to find os name using java?	java.lang.System.getProperties
19	How do I make my ArrayList Thread-Safe? Another approach to problem in Java?	java.util.Collections.synchronizedList
20	How to subtract X days from a date using Java calendar?	java.util.Calendar.add
21	Getting a list of accessible methods for a given class via reflection	java.lang.Class.getMethods
22	How to reverse a list in Java?	java.util.Collections.reverse
23	Java Convert File to Byte Array and visa versa	java.nio.file.Files.readAllBytes
24	How can I get a Unicode character's code?	java.lang.Character.codePointAt
25	Java: How to test on array equality?	java.util.Arrays.equals
26	How do I convert from int to String?	java.lang.Integer.toString
27	convert Object[] from a hashmap keyset to String[]?	java.util.Set.toArray
28	Fastest way to create new array with length N and fill it by repeating a given array	java.lang.System.arraycopy
29	How to measure time taken by Java code?	java.lang.System.nanoTime
30	Java string align to right	java.lang.String.format
31	How to split a comma-separated string?	java.util.Arrays.asList
32	toString java of arrays	java.util.Arrays.toString
33	How to find the exact word using a regex in Java?	java.util.regex.Matcher.find
34	Remove all non alphabetic characters from a String array in java	java.lang.String.replaceAll
35	Java reflection get all private fields	java.lang.reflect.Field.getType
36	How to get the current date and time of your timezone in Java?	java.util.TimeZone.getDefault
37	How can I get screen resolution in java?	java.awt.Toolkit.getScreenSize
38	Java: join array of primitives with separator	java.lang.String.join
39	Grouping elements of a list into sublists (maybe by using guava)	java.util.stream.Collectors.groupingBy
40	How to wrap text in a JTextArea	javax.swing.JTextArea.setLineWrap
41	How to convert Integer to int?	java.lang.Integer.intValue
42	Grouping by List of Map in Java 8	java.util.stream.Collectors.groupingBy
43	How to sort by two fields in Java?	java.util.Collections.sort
44	How cancel the execution of a SwingWorker?	javax.swing.SwingWorker.cancel
45	How do I format a long integer as a string without separator in Java?	java.lang.Long.toString
46	How to get system time in Java without creating a new Date	java.lang.System.currentTimeMillis
47	How to loop over a Class attributes in Java?	java.lang.Class.getDeclaredFields
48	Calculate whether an IP address is in a specified range in Java	java.net.InetAddress.getByName
49	Comparing strings with == which are declared final in Java	java.lang.String.intern
50	How to remove newlines from beginning and end of a string (Java)?	java.lang.String.trim

Figure 7. MRR and MAP based on different similarity calculation methods

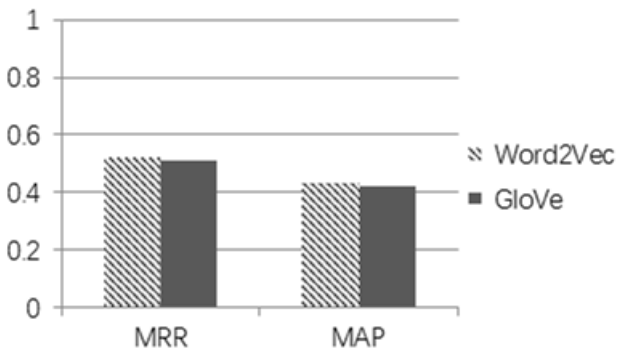


In addition, the query time is affected by the efficiency of similarity calculation in API recommended methods. The response time of the query and recommendation is determined by the similarity calculation algorithm. We recorded the time in the experiment. The average response time of WII-WMD and WV-TF-IDF is 3.6 seconds and 2.2 seconds respectively. Although WII-WMD takes 40% more time, the calculation time is acceptable based on the accuracy of API recommendations.

iv. RQ4: The word embedding technique we used is based on the Word2Vec model. There are many words embedded in the models. GloVe is one of the most popular models. We evaluated the effectiveness of API recommendation based on Word2Vec and GloVe.

First of all, 347,222 questions in the Stack Overflow data table were used as the modeling corpus. The questions are preprocessed and modeled based on Word2Vec and GloVe respectively, which are converted into the word vectors of the same dimension, that is 100 dimensions. The effective of API recommendation of two word embedding models are shown in Figure 8.

Figure 8. MRR and MAP based on different word embedding models



The API recommendation based on Word2Vec performs better than GloVe. MAP and MRR based on Word2Vec are about 2.1% and 1.9% higher than GloVe.

9. THREATS TO VALIDITY

There are some threats to the validity of API recommendation based on WII-WMD.

Our experiment datasets are from open-source projects written with Java, not considering those written with other programing languages. The top 200 questions with the Java tag are manually extracted from Stack Overflow. Theses may affect the universality of our study.

10. CONCLUSION AND FUTURE WORK

In this paper, we proposed API recommendation based on WII-WMD. Firstly, we collected the API information, such as API help document, API method, API class, API descriptions, and so on. Then API document database of API information are reconstructed and Q&A pairs including questions, answers, question ratings, answer ratings and tag information are stored in the database. Secondly, we preprocessed the collected data, including clause segmentation, deletion of stop words, stem extraction and part of speech tagging. Thirdly, word embedding modeling and TF-IDF modeling are carried out for the preprocessed text information. Then based on the new vectors, we proposed similarity calculation algorithm WII-WMD. Finally, we compared the popular calculation algorithms and verified the effectiveness of WII-WMD. Then we verified the effectiveness of API recommendation based on WII-WMD, which improves WV-TF-IDF by about 0.8% in MRR and 1.4% in MAP respectively.

Our experiment is only based on the open-source projects and Q&A on Stack Overflow is time-sensitive, so we will verify and improve our method on more datasets in the future.

ACKNOWLEDGMENT

This research was supported by Natural Science Foundation of China [grant number 61602267], the Open Project of Key Laboratories of Ministry of Industry and Information Technology for Software Development and Verification Technology of High Safety Systems at Nanjing University of Aeronautics and Astronautics [grant number NJ2018014], Outstanding Young Talents Program in Colleges and Universities of Anhui Province[grant number gxyq2017086].

REFERENCES

Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, *2*(1), 1–127. doi:10.1561/2200000006

Brandt, J., Guo, P. J., Lewenstein, J., & Dontcheva, M. (2009). *Two Studies of Opportunistic Programming: Interleaving Web Foraging, Learning, and Writing Code*. SIGCHI. doi:10.1145/1518701.1518944

Chan, W. K., Cheng, H., & David, Lo. (2012). Searching Connected API Subgraph Via Text. *Proceedings of the 20th ACM International Symposium on the Foundations of Software Engineering (FSE)*, 1-11.

Duala-Ekoko, E., & Robillard, M. P. (2012). Asking and answering questions about unfamiliar APIs: An exploratory study. *International Conference on Software Engineering*, 266-276.

Eggert, J., & Korner, E. (2004). Sparse Coding and NMF. *International Joint Conference on Neural Networks*, 2529-2533.

Frome, A., Corrado, G. S., & Shlens, J. (2013). DeViSE: A deep visual - semantic embedding model. *Proceedings of the 26th International Conference on Neural Information Processing Systems*, 2121-2129.

Goldberg & Levy. (2014). Word2vec Explained: deriving Mikolov et al.'s negative sampling word-embedding method. *ArXiv E-Prints*, 432-456.

Hoecker, A., & Kartvelishvili, V. (1995). SVD Approach to Data Unfolding. *Journal of Arxiv Cornell University Library*, *372*(3), 469–481.

Holmes, R., Walker, R. J., & Murphy, G. C. (2005). Strathcona Example Recommendation Tool. *Software Engineering Notes*, *307*(5), 237–240. doi:10.1145/1095430.1081744

Jang, , & Gu, , & Poole. (2016). Categorical Reparameterization with Gumbel-Softmax. *Italian Journal of Linguistics*, *20*(1), 33–54.

Karakasis, E. G., Amanatiadis, A., Gasteratos, A., & Chatzichristofis, S. A. (2015). Image moment invariants as local features for content based Image retrieval using the bag-of-visual Words model. *Pattern Recognition Letters*, *55*, 22–27. doi:10.1016/j.patrec.2015.01.005

Ko, A. J., Myer, B. A., & Aung, H. H. (2004). Six Learning Barriers in end-user Programming systems. *IEEE Symposium on Visual Languages & Human Centric Computing*, 199-206.

Ko, A. J., Myer, B. A., Coblenz, M., & Aung, H. H. (2006). An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Tasks. *IEEE Transactions on Software Engineering*, *32*(12), 971–987. doi:10.1109/TSE.2006.116

Krishnamurthy, B., Puri, N., & Goel, R. (2016). Learning latter-space keyword of Items for Recommendations Using Word Embedding Models. *Procedia Computer Science*, *80*, 2205–2210. doi:10.1016/j.procs.2016.05.380

Lai, S., Liu, K., He, S., & Zhao, J. (2016). How to Generate a Good Word Embedding. *IEEE Intelligent Systems*, *31*(6), 5–14. doi:10.1109/MIS.2016.45

Li, K., Chen, Y., Li, W., He, J., & Xue, Y. (2018). Improved gene expression programming to solve the inverse problem for ordinary differential equations. *Swarm and Evolutionary Computation*, *38*, 231–239. doi:10.1016/j.swevo.2017.07.005

Li, K., Liang, Z., Yang, S., Chen, Z., Wang, H., & Lin, Z. (2019b). Performance Analyses of Differential Evolution Algorithm Based on Dynamic Fitness Landscape. *International Journal of Cognitive Informatics and Natural Intelligence*, *13*(1), 36–61. doi:10.4018/IJCINI.2019010104

Li, K., Wang, H., & Li, S. (2019a). A mobile node localization algorithm based on an overlapping self-adjustment mechanism. *Information Sciences*, *481*, 635–649. doi:10.1016/j.ins.2018.12.006

Lilleberg, J., Zhu, Y., & Zhang, Y. (2015). Support Vector Machines and Word2vec for text Classification with Semantic features. *International Conference on Cognitive Informatics & Cognitive Computing*, 136-140. doi:10.1109/ICCI-CC.2015.7259377

Ma, L., & Zhang, Y. (2015). Using Word2Vec to Process Big Text Data. *IEEE International Conference on Big Data (Big Data)*, 2895-2897.

McMillan, C., Grechanik, M., Poshyvanyk, D., & Xie, Q. (2011). Portfolio: Finding relevant functions and their Usage. *International Conference on Software Engineering*, 111-120. doi:10.1145/1985793.1985809

Qin, B., Liu, T., Wang, Y., Zheng, S., & Sheng, L. (2013). Research on Chinese Question answering System based on FAQ. *Journal of Harbin Institute of Technology*, *2013*(10), 1179–1182.

Rahman, M. M., Roy, C. K., & Lo, D. (2017). Code Search in the IDE using Crowdsourced Knowledge. *Proceedings of the 39th International Conference on Software Engineering*, 51-54. doi:10.1109/ICSE-C.2017.11

Wang, F., Li, Y., Zhang, H., Hu, T., & Shen, X. (2019). An adaptive weight vector guided evolutionary algorithm for preference-based multi-objective optimization. *Swarm and Evolutionary Computation*, *49*, 220–233. doi:10.1016/j.swevo.2019.06.009

Wang, F., Zhang, H., Li, K., Lin, Z., Yang, J., & Shen, X.-L. (2018). A Hybrid Particle Swarm Optimization Algorithm Using Adaptive Learning Strategy. *Information Sciences*, *436-437*, 162–177. doi:10.1016/j.ins.2018.01.027

Wanzhi Wen received the MS and PhD degrees from Southeast University, China in 2009 and 2013. Now he is an associate professor in School of Information Science and Technology, Nantong University. His current research interests include software testing and intelligent computing.

Shiqiang Wang received the Bachelor's degree from Suzhou University,China in 2019. Now he is a graduate student in School of Information Science and Technology, Nantong University. His current research interests include intelligent computing.

BingQing Ye now is an undergraduate student in School of Information Science and Technology, Nantong University. Her current research interests include intelligent computing.

Xingyu Zhu now is an undergraduate student in School of Information Science and Technology, Nantong University. His current research interests include intelligent computing.

Yitao Hu now is an undergraduate student in School of Information Science and Technology, Nantong University. His current research interests include intelligent computing.

Xiaohong Lu now is an undergraduate student in School of Information Science and Technology, Nantong University. Her current research interests include intelligent computing.

Bin Zhang received the Bachelor's degree from Nantong University, China in 2018. Now he is a graduate student in School of Information Science and Technology, Nantong University. His current research interests include intelligent computing.