ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 5, Issue 5, May 2016

API Retrieval Method Using TFIDF Based on Cosine similarity Weighting Scheme

Shyam Kumar Pasupula, Nilam Upasani, Pushkaraj Deshmukh

***Abstract*— In this paper, we propose an Application Programming Interface (API) retrieval method using description information, in vector space model. API is the fundamental unit of work in a programming language. Large Software is explored through APIs which support the flexibility in software development process and supports reusability and frameworks. When developers invoke API methods in a framework, they often encounter a problem in finding the correct required API, from the bulk of API's in the framework. Correctly finding API in Software Development Kit (SDK) can represent a challenge. This paper aims at designing the architecture of API search engine based on TFIDF on Cosine similarity. In this paper, we also propose several key issues about the processing of Natural Language based Query Retrieval.**

***Index Terms*—API, Cosine Similarity, Information retrieval, Lucene, TF-IDF, Text mining.**

I. INTRODUCTION

In computer programming, an application programming interface (API) is a set of routine(s), protocols, and tools for building software applications. The API specifies how software components should interact with each other. A good API makes it easier to develop a program by providing all the building blocks. An API expresses in terms of its operations, inputs, outputs, and underlying types, defining functionalities. We can define an API to be the interface to a reusable software entity used by multiple clients outside the developing organization, and that can be distributed separately from environment code.

A. TF-IDF

TF-IDF stands for "Term Frequency Inverse Document Frequency", and the TF-IDF weight is a weighting scheme often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. One of the simplest ranking functions is computed by summing the tf-idf for each query term; many more sophisticated ranking functions are variants of this simple model. Tf-idf can be successfully used for stop words filtering in various subject fields including text summarization and classification.

B. Indexing:

Indexing is a crucial process in text mining. Indexing is the process of extracting text data from source documents and storing them in well-defined formatted. During the indexing process, a document is divided into its constituent units known as tokens or terms. A term is typically an individually identified key are added to the index, with the corresponding link to the document and associated Term frequency. Term frequency is the simple count of occurrence of the term in a document.

C. Information retrieval

Information retrieval refers to a process of organizing and storing up the information in a certain way, and finding them in accordance with the needs of users with the continuous improvement of the degree of information technology, information retrieval systems are used more and more widely in various fields of society.

D. Cosine Similarity

In mathematics, Cosine Similarity is a numerical statistic to measure the similarity between two vectors. Cosine similarity is defined as a dot product of the magnitude of two vectors. In the context of text mining, the Cosine similarity is used to capture the similarity between two documents represented as the angle between Query vector and the Document vector.

E. Lucene

Lucene is a powerful, high performance and scalable library for full-text indexing and searching in Java. It is suitable for application which requires full-text indexing and searching capability. Applications based on Lucene include Eclipse, JIRA, Roller, OpenGrok, Nutch, Solr, and many commercial products.

In this paper, our intention is to propose and evaluate a variety of methods that can be applied to all types of languages in the same way, through accounting for both the specificity of these search terms and on the number of terms that the query and retrieved documents have in common.

The rest of the paper is organized as follows. Section 2 describes some related work. Section 3 presents the

similarity; Section 4 focuses on the article the factors of. Section 5 focuses on the performance evaluation. Section 6 focuses on the evaluation results and Section 7 conclusion

II. RELATED WORK

In this work, the data is the form of document in which each document is described the API in the form of the name, header, input ,outputs, description of the API. Text representation, which is a fundamental and necessary process for text-based intelligent information processing, includes the tasks of determining the index terms for documents and producing the numeric vectors corresponding to the documents.

The approach mainly consists of multiple phases in that the first phase is the construction of dataset which includes generating the semi-structured data document (XML) from the unstructured API description document which presents in the header file in the commented form. This can be done by using a parser which reads the header files and generates the XML file where each API is constructed as a document which contains all basic information regarding inputs, output, and description and some of other information.

The next phase is to construct the vector space model (VSM) which is the representation of the text document in vector form. VSM was developed by Gerard Salton in SMART information retrieval system [4]. For The construction of VSM first, we need to perform pre-processing on the data which includes the removal of numbers, special characters, non-ASCII characters, tokenization, stop words, and performing streaming of tokens. The result of the result of the pre-processing will generate the BOW's (BAG of Words). The union of BOW gives the dimensionality of the problem space. This vsm is used by the search engine to find the similarities between the vectors in the form of terms or tokens appear in the document.

The next phase is to construct the term document matrix. The Term-Document Matrix is an array of vectors where the columns represent the tokens in the document and each row as a document. The term document matrix stores the frequency of the term of all token in the document.

The next phase is to construct the TF-IDF matrix which is used to find the importance of the token (word) in the data set, handling of stop words and also perform the normalization which will overcome the problem of multiplication of zero. In phase, we use to calculate the TF-IDF value of each cell in the matrix.

The next phase is to calculate the cosine similarity. In mathematics, Cosine similarity is a numerical statistic to measure the similarity between two vectors. Cosine similarity is defined as a dot product of the magnitude of two vectors. In the context of text mining, the Cosine similarity is used to capture the similarity between two documents represented as the angle between the query vector and the document vectors. The cosine similarity will give better score then as the Euclidean distance. Euclidean distance only finds the similarity in distance, whereas cosine will give both the distance and the direction of the vector.

III. SIMILARITY

The system generates the Term-Document Matrix which is the concatenation of the all document vector in the form of two dimension matrix, where each row represents the document in the data set. The field in the vector of a document is BOW's. The value of each cell in the term vector matrix represents the word appears in the respective document.

The next we need to calculate the tf- idf value of each cell in the matrix which is used to normalized the Term-Document Matrix. The formula for calculating the tf-idf is as follows.

TF-IDF = TF (termcount) * IDF (docfreq, numberdoc);
TF (termcount) = $\log_{10}$ (1+ termcount);
IDF (docfreq, numberdoc) = $\log_{10}$ (numberdoc/ (docfreq));

The TF (Term Frequency) is used to normalize the count of the term. The formula which is used by TF function is to handle the zero exception also been handled.

The IDF (inverted document frequency) which is used to find the weighting for the token in the data set which is nothing but the fraction of the number of documents by the word appeared in a number of the document. The value which is generated by the function is always greater than one. The weight is high when the word appears less in documents had the more weight the weight is always between the $\log_{10}$ (1) to $\log_{10}$ (#documents) respectfully minimum and maximum weight for a token.

The new VSM matrix consists of the normalized weighted values in each cell. The value of the cell is calculated by the product of TF*IDF value with respect to document and the token weight.

The cosine similarity is nothing but the value of cos angle between the two vectors. The cosine value always lies between the 0 to 1 which is always inversely proportional to the angle mean the more the angle, less cosine value. The cosine is nothing but the dot product of two vectors.

The formula for calculating the cosine value of two vectors of 'n' dimension is as follows.

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Example consisted the two vectors let A,B are two doc vector and search query vector. Let the A had the vector

values like [0.1, 0.3, 0.9] and the B had the vector values like [0.2, 0.5, 0.4].

$$A.B = a1*b1+a2*b2+..;$$

$$\|A\| = \sqrt{(a1*a1+a2*a2+...)};$$

Hence,
A.B= (0.1*0.2) + (0.3*0.5) + (0.9*0.4) = 0.53.
$\|A\|=\sqrt{(0.1*0.1+0.3*0.3+0.9*0.9)} =\sqrt{(0.91)} = 0.953.$
$\|B\|=\sqrt{(0.2*0.2+0.5*0.5+0.4*0.4)} =\sqrt{(0.45)} = 0.67.$
Cos (θ) = 0.53/ (0.953*0.67) = 0.83.

The value cosine determines the angle between the vectors. If the value of the cosine is 0 it means that the vectors are orthogonal (90˚) to each other. If the value of the cosine is 1 it means that the vectors are similar means the angle is 0˚ not all terms may or may not contained in the vectors. If the value of the cosine is >0, and <1. Then the angle is the acute angle between the vectors means they are not similar but have some common terms.

IV. ARCHITECTURE

The approach mainly contains 6 modules like dataset creation, pre-processing the data, indexing, storing the indexed documents, query builder, and searcher.

A. Dataset

Any text-based system requires some representation of documents, and the appropriate representation depends on the kind of task to be performed. Different from data mining that handles the well-structured data, text mining deals with a collection of unstructured documents without any special requirements for their composition except some general grammar and lexical rules. This makes that one of the main themes supporting text mining is the transformation of text into numerical data, i.e., text representation.

Text representation, which is a fundamental and necessary process for text-based intelligent information processing, includes the tasks of determining the index terms for documents and producing the numeric vectors corresponding to the documents. In this paper, We use the traditional indexing method as TF*IDF (Term Frequency Inverse Document Frequency).

In this work, the data is the form of document in which each document is described the API in the form of the name, header, input ,outputs, description (functionality) of the API. Description of Creo API's is available on the headers in commented format.

```
Return_type APIname  (args);
/*DEPRECATED: <version>
SUCCESSORS: <new API name>
Purpose: <Api funcationality or description >
Input Arguments:
arg       - description.
Output Arguments:
arg       - description.
Return Values:
ERROR     - ERROR description.*/
```

The unstructured data is to be converted to structure format for that we parse all the header file of the Creo and structured into an XML document.

```
<doc>
<method\>
<header_type\>
<Deprecated\>
 <Successors\>
 <License\>
<header\>
<arguments\>
 <Method_Description\>
<Input_Description\>
<Output_Description\>
 <return\>
</doc>
```

B. Pre-Processing

After extracting the API information into a structured XML document the next step is to perform some data cleaning which includes the removing of comments and tags in the description, converting to lowercase, stop-words removing and stemming.

The text extracted from the source contain the unnecessary elements like NON-ASCII characters, tags and comments element like ['//', '/*', '*/']. Which is the noise in the data is removed.

Next, the data is to be converted to lowercase. The step is performed to normalize the text for making the keyword match case insensitive.

Stop words are basically a set of commonly used words in any language, not just English. The reason why stop words are critical to many applications is that, if we remove the words that are very commonly used in a given language, we can focus on the important words instead and increase the performance of the system.

Stemming is a pre-processing step in Text Mining applications as well as a very common requirement of Natural Language processing functions. The main purpose of stemming is to reduce different grammatical forms / word forms of a word like its noun, adjective, verb, adverb etc. to its root form. We can say that the goal of stemming is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.

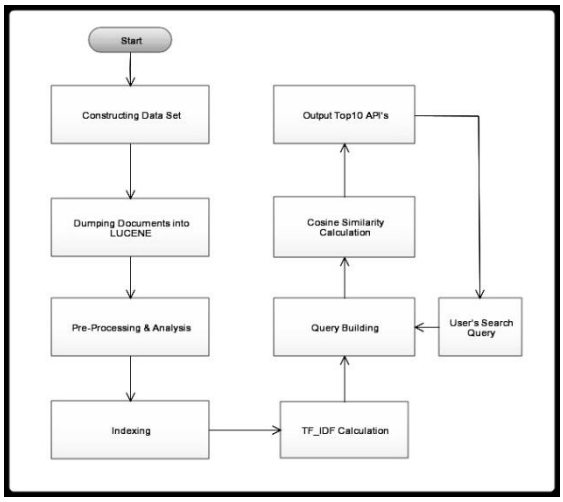

Fig 1 System Architecture

C. Indexing

Among various indexing strategies, we use of inverted file indexing is well suited for large documents collection. In simplest form an inverted file index provides a mapping of terms, such as words to its location in a text document, a document can be thought of as a collection of m terms.

The document is a collection of the field in XML file format to perform access and performance issues. In the process of indexing, this document is used to create a VSM. In this work, we dump all the information in Lucene document. Which store the key as the method name, and method description.

This method description has then performed some operation which includes which includes constructing the vector of each document.

The indexer had the sub-module which includes to performing the analysis the document. Like tokenization which is used to convert the word into tokens. Which include the dividing of the document into atom units which is a word, and then the analyzer removes the tokens which include symbols, numbers, and special charters in the token.

Next step of the indexer is to convert all the tokens into lowercase which make the normalized tokens for all the cases which improve the performance and accuracy of the system.

Then we get a bag of words nothing but Typically a document is made up of a sequence of n unique terms such that n <= m. The number n is usually far less than m as most of the terms is repeated while forming a document. For instance, the term "the" is repeated many times in this paper. The set of unique terms within an index forms the "Term List" v of the index. If a pointer (say numeric location) is associated with each term in v to the location of that term in a text document, the resultant data structure is a form of inverted file index. As the document collection grows, the number of documents matching a term in the index becomes sparser.

The index is oftentimes annotated with the information regarding the frequency of occurrence of each term in the document. The representation of a document as a vector of frequencies of terms is referred to as *vector space model*.

V. EVALUATION

We conducted an evaluation to assess the effectiveness of the system. The experiment is mainly used to find the system efficiency as the system not only API query retrieval engine it also supports the synonyms free, stemming, unordered of API description. To check perform of the system. We used to evaluate the performance of methods which is to check the system efficiency.

The performance of the system is mainly done by creating the test set which contains 10% of total number of API in the system. In which the test tuple are selected by randomly for the data set of all API's. The test set contains the key, the value was key the query which is the description of API and the value is the API name which relevant of the description. This test data set is sent to the system and the run each and every query and print the TOP 10 matches for the relevant system and the results are dumped into the XML file the capture each query results in a top match first up to top 10 lists into the single doc for each query with their query values.

The result of the system is being observed as that the system is fetching the output as 100% efficient in the performance case as the statement is description is same when the user queries are passed into the system the efficient is not as much as the expected being the data is not having more and properly documented and the less information is being availability. The experiment is mainly done on a property framework as internship project with the *Parametric Technology Corporation* (PTC) inc, pune. PTC had a framework of c & c++ framework. In the framework, we take a toolkit department related API's which are documentation is done.

The data set of the department contains the 7K+ API's of both c and c++ language in that some of them are not documented and some are having a lack of information in the description.

We take this data set to test the system performance in both procedures that is as create the data into XML format and passing to the system to build vector space model. Then we construct a test data set using Systematic random sampling in which we have to take every 10^{th} element for the list of data set and constructed the test data set.

VI. RESULTS

As we mentioned precisely that our performance is carried by which is from the construction of the same test data set from the training data to test the system .the test data is the 10 percent of the total training dataset. As we seen performed the evaluation which is mentioned above. The result of the first performs method we get the 100 correctly classified apis.

VII. Conclusion

In this paper, we present a document semantic similarity calculation method that combines cosine similarity measure and TF-IDF based weighting scheme which is used to find the API based on question and answer style. The method which gives best results when the information of the document is full-fledged.

Further research can be done on processing the system answer user not only based on static information of question but also supports to hand the semantics of the question and feedback processed system to stabilize the system.

VIII. Acknowledgment

This work is mainly done for PTC creo object toolkit API framework, which is an object-based c++ language framework for ptc creo parametric and PTC Creo direct. PTC Creo toolkit makes possible the development of C++ programs that access the internal components of a PTC Creo session, to customize PTC Creo models.

IX. Reference

[1]. Martin P. Robillard, Eric Bodden, David Kawrykow, Mira Mezini, and Tristan Ratchford. "Automated API Property Inference Techniques" IEE TRANSACTIONS ON SOFTWARE ENGINEERING vol. 39, no. 5, May 2013.

[2] .Rahul Pandita , Raoul Praful Jetley, and Sithu D Sudarsan , Laurie Williams "Discovering likely mappings between APIs using text mining" In Proc. Source Code Analysis and Manipulation (SCAM), 15th IEEE International Working Conference on 27-28 Sept. 2015.

[3].Shouning Qu ,Sujuan Wang,Yan Zou. "Improvement of Text Feature Selection Method based on TFIDF". International Seminar on Future Information Technology and Management Engineering, 2008.

[4].Mohammad Alodadi and Vandana P. Janeja ."Similarity in Patient Support Forums Using TF-IDF and Cosine Similarity Metrics". International Conference on Healthcare Informatics, 2015

[5] .Yanzhen Zou, Ting Ye, Yangyang Lu1, John Mylopoulos, Lu Zhang. "Learning to Rank for Question-Oriented Software Text Retrieval", 30th IEEE/ACM International Conference on Automated Software Engineering,2015.