INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X

(Volume2, Issue6)

Available online at: www.Ijariit.com

Implementation of Plagiarism Detection Tool in JAVA

Attinderpal Singh

CSE, Guru Nanak Dev University, Amritsar

attinder.saini@gmail.com

Abstract— with the advent of internet everything that has been researched is available on the internet. This information can be copied and used as original by someone in their own work which poses a major challenge for academia and industries like publications. A lot of work has been done in the field to detect plagiarism in the documents and algorithms have been made. In this paper, we have implemented porter algorithm for stemming, TF-IDF and cosine similarity to detect plagiarism.

Keywords— plagiarism, porter, TD-IDF, cosine, similarity, java.

I. INTRODUCTION

Plagiarism is defined as to steal someone's ideas or words claim as their own. It can also be defined as to copying the data from another source and present them as their own findings. In other words if someone uses other person's work without his permission and present it under his own name without full acknowledgement is considered as an act of plagiarism[3]. This type of stealing is also known as copyright infringements or text reuse.

The thoughts of every person are unique. Two people can't write exact words even if they can speak the same language. In earlier times, the plagiarism was not easy to detect because it was done manually. The manual approach was not very efficient and it was hard to detect plagiarism because it was completely based on the person's awareness about the text.

Plagiarism has become the problem for the academic and industries like publication that publish the work of another person and also educational institutions.

A simple way to avoid plagiarism if we have to include someone else's work in our work is done by citation of the source. If a person cites the source, then person is allowed borrowing the ideas or phrases from the sources[3]. If our work prevails over someone else's ideas or words then it is considered as plagiarism. If we use the words of the sources too closely then it becomes the case of plagiarism. Therefore we should be very careful while citing the sources.

II. DIFFERENT TOOLS FOR DETECTING PLAGIARISM

Due to extensive research in this field there are number of plagiarism detection tools developed. There are paid as well as free services available online. Grammerly, White smoke, Dust ball, Plagspotter are some examples. These tools take the original document and verify it against its inbuilt database or across the web to check if it is copied. All of these are popular tools and are efficient in detecting plagiarism. These tools search over the internet like most of the search engines with set of algorithms to find the similarities in the text. The pages which has more unique content has higher rank and which has duplicated content has lower rank. Indexes used by these tools for searching are provided by its own database.

III. AVAILABLE METHODS

There are different approaches to detect plagiarism in documents. Fingerprinting is currently most widely accepted approach to detect plagiarism [7]. Multiple substrings are matched to suspicious document and if the exceed a certain threshold it is considered plagiarized. String matching is another technique in which pair of stings is matched for all the documents in the reference collection. It remains computationally expensive task. Citation based plagiarism detection (CbPD) relies on citation analysis. It does not rely on textual similarity [7]. It compares the citation and reference information to identify similar patterns. It is relatively new concept. Another method is Stylometry which subsumes the author's unique writing style.

IV. TECHNIQUES

A. Tokenization

Tokenization is the process of breaking a stream of text into words, phrases, symbols, or other meaningful elements called tokens. The aim of the tokenization is the exploration of the words in a sentence. The list of tokens becomes input for further processing such as parsing or text mining. After tokenization non words are removed such as symbols like '@','#','&',""'.'?' etc.

Stop word removal

Many words in the documents recur frequently but are meaningless as they are used to join words in a sentence [4]. They do not contribute to the context or of the document. Due to their higher occurrence their presence is an obstacle in text mining. Stop words which are very frequent are 'and', 'this', 'are' etc. They must be removed because they are not useful in classification of documents. In English literature there are 400-500 stop words [4].

Synonym Removal

Synonyms are removed from each document using Word Net library. For e.g. if a text contains "Hello", "Hi", "Hullo", "Howdy" then all are replaced with "Hello".

Stemming

Stemming can be defined as chopping off the ends of the words and hoping to achieve similar context most of the time. It converts the words in the document to their stems or root words. The reason being most of the times the words with the same stem describe the same meaning in the text, so words can be conflated together by their stems.

Rules	Examples
SSES →SS	PASSES → PASS
S →	DOGS → DOG

In the present work, the Porter Stemmer algorithm, which is the most commonly used algorithm in English, is used [4].

V. Porter Algorithm

Porter Algorithm is used to stem the suffixes. A suffix stripping program is provided with a list of suffixes and conditions under which it may be removed so that the stem remains valid under the context of the sentence. There are some problems within the algorithm like it conflates words like RELATE and RELAITIVITY which in physics context might cause problems but adding more rules in vocabulary for performance in one area means degradation in the other area [1]. Moreover root words which change after adding suffix are rarer in real vocabulary than one might expect. So in that it is not worthwhile to handle these cases.

From every word of every sentence in every document, porter algorithm is applied to convert every word to its root. This input is further passed to calculate TF of every word.

VI. Tf-idf Calculation

A. Term Weighting

In the vector space model, the documents are represented as vectors. Term weighting is an important concept which determines the success or failure of the classification system [4]. Two major indicators of level of importance of words in the text are Term Frequency (TF) and Inverse Document Frequency (IDF). In TF we assign a number as a weight to the term and that number is equal to the number of occurrences of term in the document. This constitutes a problem because all terms are considered as equal in TF. So IDF is used to emphasize and provide more weight to the less frequent words which provide more contexts to the document. TF/IDF is a technique which uses both TF and IDF to determine the weight a term [4].

B. Term frequency

If we have a query to determine which document is most relevant to the query. A basic approach is to start eliminating all the documents which do not have the words given in the query. By further analysis we can count the number of times the words in the query are recurring in the document and distinguish between the documents. That count of terms is called its term frequency.

In the case of the term frequency TF(t,d), the simplest choice is to use the raw frequency of a term in a document, i.e. the number of times that term t occurs in document d. If we denote the raw frequency of t by $f_{t,d}$, then the simple TF scheme is TF(t,d) = $f_{t,d.}$ [5]

C. Inverse Document Frequency

If a term is very common in the document then term frequency will incorrectly focus on the documents which are using that term. Example of such a term is 'the'. A document which has a lot of 'the' in it will emphasize document which has term 'the' in it without giving enough weight to more meaningful words like 'white' and 'dog'. The term 'the' is not a good keyword to check the relevancy of terms in the document. So an inverse document frequency is used which reduces the weight of terms that occur very frequently in the documents and increases the weight of terms which occur rarely[2].

For e.g.: "play" word appears only in Document2

IDF(play) = 1 + loge(Total Number Of Documents / Number Of Documents with term game in it)
There are 3 documents in all = Document1, Document2, Document3
The term play appears in Document1
IDF(play) = 1 + loge(3 / 1)
= 1 + 1.098726209
= 2.098726209

D. Duplicate Removal

After TF calculation, duplicate words are removed from each sentence from every sentence. For e.g. "a man eats fruits man and vegetables."

This sentence is reduced to "a man eats fruits and vegetables".

E. TFIDF Vector Space Model

The TF IDF weight of a term is the product of its TF weight and IDF weight. It is often used in information retrieval and text mining. The class of weighting schemes known generically as TF*IDF, which involve multiplying the IDF measure (possibly one of a number of variants) by a TF measure (again possibly one of a number of variants, not just the raw count) have proved extraordinarily robust and difficult to beat, even by much more carefully worked out models and theories[2].

The Dot Product

The definition of the dot product is a simple multiplication of each component from the both vectors added together. The example of a dot product for two vectors with 2 dimensions each (2D):

vec{a} = (0, 4)
vec{b} = (5, 0)
vec{a} .vec{b} = 0*4 + 5*0 = 0

F. Cosine Similarity

The cosine similarity is a measure to calculate cosine of the angles between to vectors or two documents on the vector space. This is used for judgement of orientation not magnitude. In information retrieval and text mining each term is assigned different dimension and document is marked by a vector and the value of each dimension is number of times that term appears in the document. What we have to do to build the cosine similarity equation is to solve the equation of the dot product for the cos (theta)[6]:

cos(theta)=(vec{a}.vec{b})/(|vec{a}|*|vec{b}|)

Two vectors of same orientation has cosine similarity of 1, at 90 degrees 0 and two vectors which are opposite have a similarity of -1. It is particularly used in positive space. In the case of information retrieval it has range from 0 to 1 because TF-IDF cannot be negative. Even if we have a vector pointing to a point far from another vector, they still could have an small angle. The measurement tends to ignore the higher term count on documents. Suppose we have a document with the word "blue" appearing 100 times and another document with the word "blue" appearing 10, the Euclidean distance between them will be higher but the angle will still be small because they are pointing to the same direction, which is what matters when we are comparing documents.[6]
Cosine similarity of every sentence from the test document is found with every sentence of each training dataset document. If the value which ranges from 0 to 1 exceeds certain threshold value for e.g. 0.75 then the sentence is said to be plagiarized i.e. copied.

FIGURE 1: Plagiarism check tool

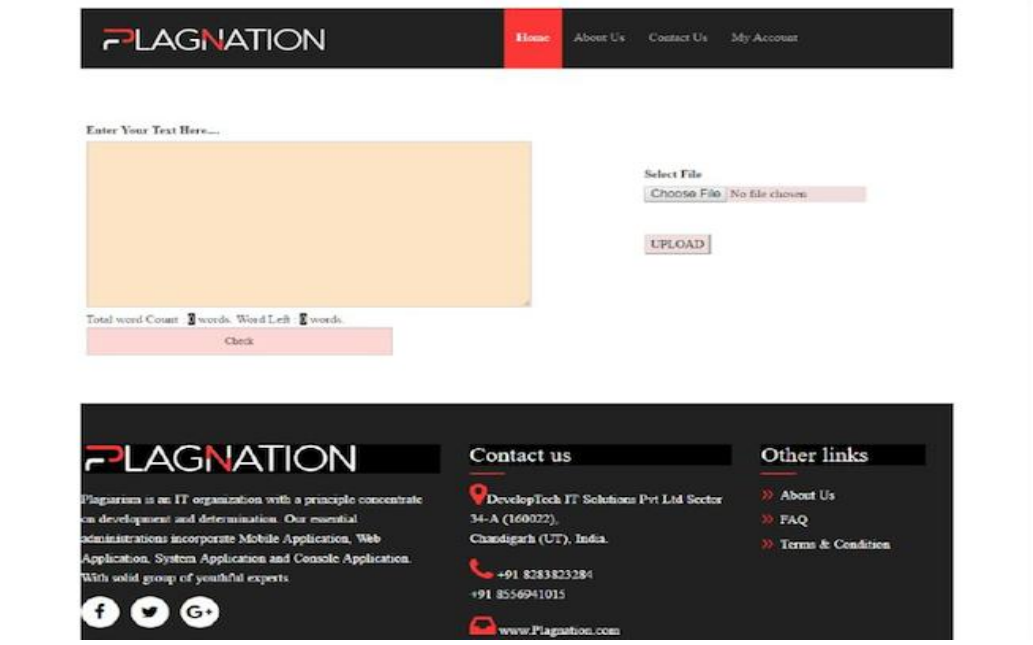

FIGURE 2: Upload File to check plagiarism

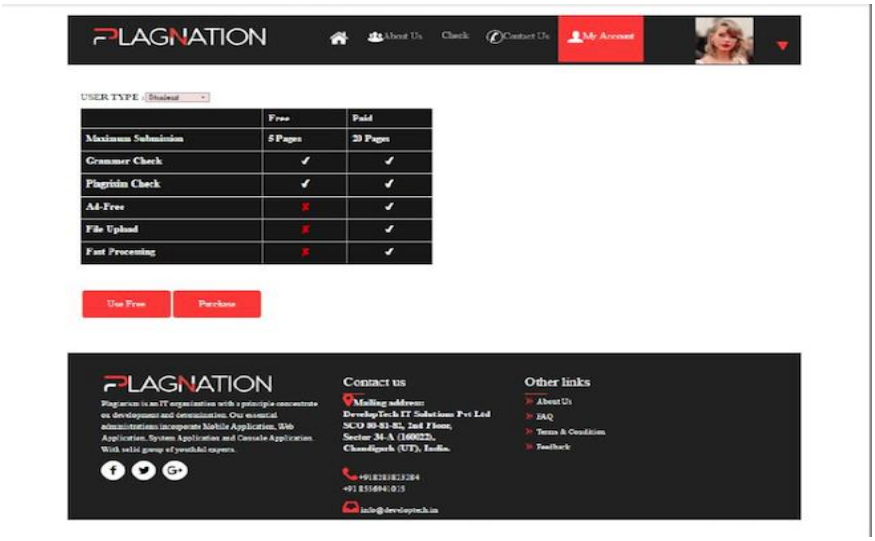

FIGURE 3: PERCENTAGE OF COPIED CONTENT

REFERENCES

[1] Porter, Martin F. "An algorithm for suffix stripping." *Program* 14.3 (1980): 130-137.

[2] Robertson, Stephen. "Understanding inverse document frequency: on theoretical arguments for IDF." *Journal of documentation* 60.5 (2004): 503-520.

[3] What is plagiarism. http://www.plagriarism.org/

[4] Srividhya, V., and R. Anitha. "Evaluating preprocessing techniques in text categorization." *International journal of computer science and application*47.11 (2010).

[5] Tf-idf https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[6] Christin S. Perone, Machine Learning: Cosine Simlilarity for Vector Space Models part (III). http://www.citeulike.org/user/DenvildegruppeP3/article/13886247

[7] Plagiarism Detection, December 2010. Retrieved from http://en.wikipedia.org/wiki/Plagiarism_detection - Detect ion_methods.

