*Al-Azhar University-Gaza*

*Deanship of Postgraduate Studies*

*Faculty of Engineering & Information Technology*

*Master of Computing and Information Systems*

# Intelligent Plagiarism Detection for Electronic Documents

**By:**

**Mohran H. J. Al-Bayed**

**Supervisor**

**Prof. Dr. Sami S. Abu Naser**

Professor of Artificial Intelligence

**A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master in Computing and Information Systems**

October – 2017

جامعة الأزهر – غزة

عمادة الدراسات العليا

كلية الهندسة وتكنولوجيا المعلومات

ماجستير الحوسبة ونظم المعلومات

# الفحص الذكي للاقتباسات العلمية في الوثائق الإلكترونية

**إعداد:**

**مهران حموده جمعة البايض**

**إشراف**

**ا.د. سامي سليم ابو ناصر**

استاذ دكتور الذكاء الإصطناعي

**قدمت هذه الرسالة استكمالاً لمتطلبات الحصول على درجة**
**الماجستير في الحوسبة ونظم المعلومات**

محرم – 1493

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ

صَدَقَ اللهُ العَظِيم

# DECLARATION

I declare that I have written this thesis and that this work has not being submitted -as a whole or any part of them- for any other degree or professional qualification or any other.

I confirm that this work that submitted is my own, except where work which has formed part of jointly-authored publications has been included. My contribution and those of the other authors to this work have been explicitly indicated below.

I confirm that appropriate credit has been given within this thesis where reference has been made to the work of others.

*Mohran Hammouda Joumaa Al-Bayed*

# DEDICATION

This research is dedicated to:

To My Father's Soul...

To My Mother for Their Love, Pray, And Continuous Sacrifices...

To All of My Brothers and Sisters and Their Kids...

To My Wife...

To My Friends, Colleagues and My Teachers...

With All My Love.

**Mohran Hammouda Joumaa Al-Bayed**

# ACKNOWLEDGEMENTS

All praises and much gratitude to almighty Allah, the most compassionate and magnificent, who gave me the power to work hard and persistence to complete this research work.

I would like to specially thank my supervisor **Prof. Dr. Sami S. Abu Naser**, who always provided me with the greatest support and help whenever I needed throughout my research work. He was readily available for consultation, shared his knowledge with me as well as motivated me for successful completion of this research work.

I cannot forget the successful support of my beloved parents, my brother Mohannad, my sisters, and my wife Yasmeen, who always have shown desire and prayed for my success in my academic life in addition to the financial and honorable support through my life.

I would like to thank all my reputable professors, honest friends and all those people in the Faculty who aided me throughout this research project and made this thesis blossom.

***Mohran Hammouda Joumaa Al-Bayed***

# ABSTRACT

Plagiarism detection is the process of finding similarities on electronic based documents. Recently, this process is highly required because of the large number of available documents on the internet and the ability to copy and paste the text of relevant documents with simply Control+C and Control+V commands.

The proposed solution is to investigate and develop an easy, fast, and multi-language support plagiarism detector with the easy of one click to detect the document plagiarism. This process will be done with the support of intelligent system that can learn, change and adapt to the input document and make a cross-fast search for the content on the local repository and the online repository and link the content of the file with the matching content everywhere found.

Furthermore, the supported document type that we will use is word, text and in some cases, the pdf files –where is the text can be extracting from them- and this made possible by using the DLL file from Word application that Microsoft provided on OS. The using of DLL will let us to not constrain on how to get the text from files; and will help us to apply the file on our Delphi project and walk throw our methodology and read the file word by word to grantee the best working scenarios for the calculation.

In the result, this process will help in uprising the documents quality and enhance the writer experience related to his work and will save the copyrights for the official writer of the documents by providing a new alternative tool for plagiarism detection problem for easy and fast use to the concerned Institutions for free.

**Keywords:** *Plagiarism Detection, Intelligent System.*

# الملخص

فحص الاقتباسات في البحوث العلمية هو عملية إيجاد أوجه التشابه داخل الوثائق الإلكترونية. هذه العملية مطلوبة للغاية في الآونة الأخيرة بسبب العدد الكبير من الوثائق المتاحة على شبكة الإنترنت وامكانية نسخ ولصق النص من الوثائق ذات الصلة بالموضوع بسهولة الضغط على كونترول + C وكونترول + V.

هذا البحث يقدم حل مقترح لتطوير طريقة سهلة وسريعة لكيفية فحص أوجه التشابه في الوثائق الإلكترونية، مع اضافة دعم العديد من اللغات للبرنامج مع العمل على فحص الاقتباس في داخل الوثائق الالكترونية بسهولة وبضغطة واحدة. ستتم هذه العملية بدعم من النظام الذكي الذي قمنا بتطويره والذي يتعلم ويتغير ويتكيف مع الوثيقة المدخلة ويقوم بإجراء فحص سريع للمحتوى على قاعدة البيانات المحلية وعلى مواقع الانترنت مع مطابقة المحتوى في كل مكان يتم ايجاده.

من ناحية اخرى، أنواع الملفات المدعومة من قبلنا تتنوع ما بين ملفات الوورد، النصوص وفي بعض الاحيان ملفات pdf –التي يمكن استخراج النصوص منها– وهذه العملية تتم بمساعدة ملفات DLL التي يقدمها برنامج الوورد التي تطوره مايكروسوفت المقدم على نظام التشغييل الخاص بها. استخدام ملفات الـ DLL سيجعلنا لا نقيد بكيفية استخراج النصوص من الملفات; وهذا سيساعدنا في استخدام الملف في تطبيقنا المطور ببرنامج الدلفي باستخدام حلنا المقترح لضمان الحصول على افضل سيناريوهات الحسابات.

بالتالي سوف تساعد هذه العملية في تحسين جودة الوثائق الالكترونية وستقوم بتعزيز تجربة الناشرين والباحثين المتعلقة بالبحوث والوثائق العلمية، وسوف تقوم بحفظ حقوق الملكية للكاتب الرسمي للوثائق عند ايجاد اي تشابه بالوثيقة  بايجاد طريقة بديلة بسرعة وفعالية لفحص الاقتباسات العلمية وتقديمه للمؤسسات المعنية بطريقة مجانية.

**كلمات مفتاحية :***فحص الاقتباسات العلمية، نظام ذكي.*

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF EQUATIONS

# PROJECT IN BRIEF

| | | |
|---|---|---|
| **PROJECT TITLE** | : | INTELLIGENT PLAGIARISM DETECTION for ELECTRONIC DOCUMENTS |
| **UNIVERSITY** | : | Al-Azhar University - Gaza |
| **SUBMITTED BY** | : | Mohran Hammouda Joumaa AlBayed |
| **SUPERVISED BY** | : | Prof. Sami Abu Naser<br>Professor of Artificial Intelligence,<br>Faculty of Engineering & Information Technology,<br>Department of Information Technology,<br>Al-Azhar University - Gaza. |
| **TOOLS USED** | : | Delphi 2010(for development purpose)<br>SuperObject (A lightweight library to convert Objects into JSON and back)<br>PerlRegEx (for regex evaluation on the sentence)<br>DIHtmlParser (for querying HTML documents)<br>DIDuckDuckGoReader (A DIHtmlParser plugin specially optimize to retrieve data from Duck Duck Go search engine)<br>MS Office 2017 DLL with Delphi (for retrieving documentation & presentation) |
| **OPERATING SYSTEM** | : | Windows 10 (64-bit.) |
| **START DATE** | : | January, 2017 |
| **COMPLETION DATE** | : | October, 2017 |

# Chapter 1

# INTRODUCTION

# 1. Introduction

This chapter summarizes the information that will let you emphasize the meaning of plagiarism in a simple meaning.

## 1.1. Plagiarism

Early in the 17th century, The word *Plagiarius* recorded; *Plagiarius* is the Latin source of the word plagiarism; it is defined as "*The exercise of taking somebody else's effort or thoughts and passing them as his/her own*" [1]. Furthermore, this word comes from Latin plagiarius '*kidnapping'*.

In this Era, the huge and fast evolution on the technology's and the new data available is increasing every day. That's meaning in the simplest way of this fact we will have millions of documents that are available online and this lead to the possibility to take some parts –or whole maybe- form any documents of them and the ability to copy and paste the text of relevant documents with simply Control+C and Control+V commands. Therefore, copying from others sources, statements or even talks in your document without notifying that parts are from others is called Plagiarism.

Meanwhile with these different sources of information and documents, this process of detection is very important and get very harder every day, regarding to the highly impact of this process in the educational level. Therefore, we need to find a solution that make the educational institution guarantee that work is not belong to others and save rights of original author and source.

### 1.1.1. Plagiarism Detection

This complicated process increasing over time even in the higher levels of education. Hence, we always need to find and detect this case within the document as described in Figure 1.1 and this can be done with this detection techniques:

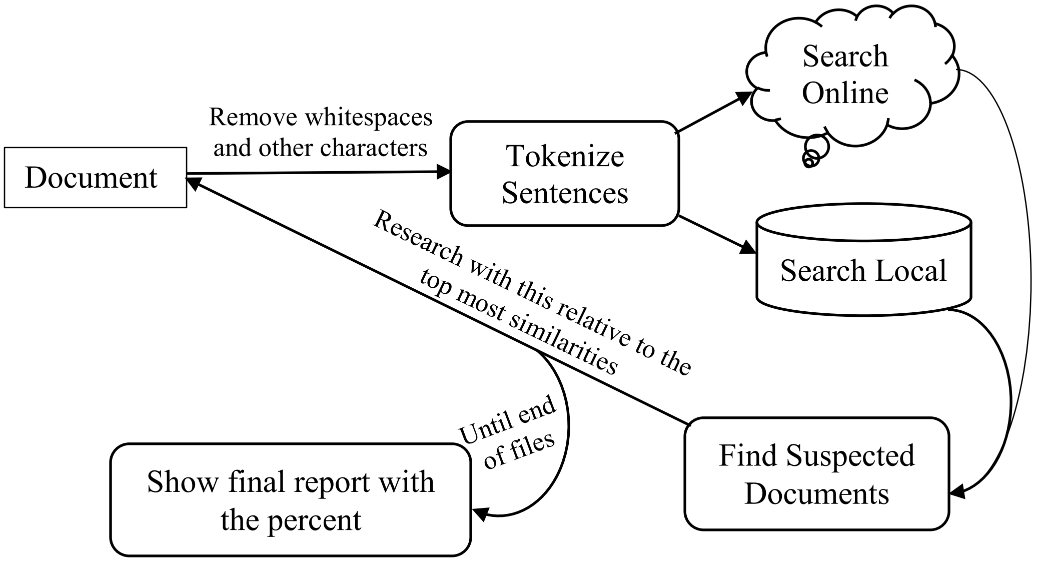

Document
Remove whitespaces and other characters
Tokenize Sentences
Search Online
Search Local
Find Suspected Documents
Research with this relative to the top most similarities
Until end of files
Show final report with the percent

*Figure 1.1: the simple procedure of the Plagiarism Detection*

### 1.1.2. Intrinsic Plagiarism Detection

If no reference of the document available how can we check the plagiarism? This recent method of detecting is used to detect the text pieces, sentences or even a block of text copied as whole part even without any external knowledge. This process can be made by detecting changes inconsistencies within a given document [2]. Another solution is by using Vector Space Models [3] that use a few subjects independent stylometric characteristics from which a vector space model for every sentence of a suspicious document is built, or even by using Complexity Analysis [4] that use Kolmogorov Complexity measures as a method of digging out structural data from the manuscripts for Intrinsic Plagiarism Detection.

### 1.1.3. Extrinsic Plagiarism Detection

When we have knowledge about the suspicious references of the files that the author maybe plagiarized from, we use this method with highly dependent of the suspicious and check if we found matches in keywords, sentences or even whole blocks. This process can be made by using Fuzzy Semantic Based tokenization [5] of the string similarity and search in a list of suspected documents and find their similarity. Another solution is by using cross-language semantic textual similarity detection [6] by using the Cross-Language Character N-Gram [7] typically by configuring the document and tokenizing the sentences which break words at spaces, downcast them and remove diacritics (◌ْ◌ّ◌ٍ◌ِ◌ٌ◌ُ◌ً◌َ) to identify sentence boundaries to improve accuracy. Alternatively, by using the Cross-Language Conceptual Thesaurus-based [8] that measures the distance between sentences and the possible translation of each word in them, and evaluated to each sentences possible translation. In the same way, the Cross-Language Alignment-based Similarity Analysis [9] that are aims to find the similarity between sentences and the translation that are found in bilingual unigram dictionary which contains translations pairs (and their probabilities) [10] that are already generated by using high performance computers. Another possible solution is by using the Cross-Language Explicit Semantic Analysis [11] that compare the documents by using interpretation vectors that are a weighted vectors of concepts based from the translation derived directly from the Wikipedia.

## 1.2. Research Objectives

This research explores the available methodology to detect plagiarism on the documents, especially on the field of science.

In general, the way to detect plagiarism on document is to tokenize the files into a number of tokens, search for them on other files, and find the matching among them. The fundamental issues are examining any document carefully:

- Find best plagiarism detection method to use.
- Improve detection through multiple experiments with the help of real documents and users.
- Search the same file with relative plagiarism system to find the best match.
- Provide a full indexed reference of the parts that were plagiarized.
- Provide the Multilanguage support of mean of use for English and Arabic documents.
- Provide a full reference for the researcher how to get the best result when use our system for detecting plagiarism.
- Introduce the system to our university for free to help to enhance the research quality in our universities.

Moreover, this lead us to our main objective, which is providing a new alternative tool for plagiarism detection problem by providing a new alternative tool for easy and fast use to the concerned Institutions for free..

## 1.3. Research Limitations

We need to pre-process the file to remove any unwanted text from it such as Punctuation marks and Diacritics from the text. In the other hand, we cannot search more than one file per time to take the full advantage of the speed search that can be solved by using parallel computing, and the last one is the problem of changing techniques on the web search and their search engine optimization over time that required some minor modifications in the code. This can be possible by letting the user to modify the syntax of the document to be the same as the site.

## 1.4. Problem Statement

For efficient and fair plagiarism detection, we need to check the document with the existence documents published online over the web. A recent research published from the University of Ottawa [12] have shown that approx. 2.5 million of science published documents that are relevant to a lot of topics with a mainly 4-5% increasing each year. How can we make an efficient way to find the matching with millions of documents that are publishing each year?

If we assume each file will make about 1 Megabyte then we have 2.5 million of Megabyte, i.e. 20 Terabyte of increasing storage per year. How can we handle the huge repository size of documents during plagiarism detection? Do we need to store this file for future search?

Our system will introduce a solution for these problems with the intelligent feature that can learn and optimize the detection to its minimum cost and the highly quality result. This will be possible by searching the document using different search engines like Google [13], DuckDuckGo [14] or any search engine that provide a flexible search feature without the need to store this files on our local storage. This made us very motivated to find a new way to detect plagiarism with the help of external detection mechanism.

## 1.5. Overview of The Methodology

Our system methodology consists of the following:

1. Pre-processing the file and remove any Punctuation marks, Diacritics and remove any special character like character formation in Arabic Language.
2. Read text word by word, this will be using the help of mathematical Regex and Tokenize the words based on a fixed sliding window of text that can be changing by the user.
3. Search for the token-sliding window- over the web; download the result and extract the exact result for the search and calculate the token plagiarism percentage.
4. Generate the suspected list to enhance result gathering.
5. Loop throw tokens until final token as same before.
6. Calculate the major token plagiarism percentage for the whole file and prepare the report with feedback needed to the researcher or university assistance.

## 1.6. Structure of The Thesis

In this thesis, we have 5 chapters. The first chapter will constrain on what is the

Plagiarism and how to detect that case in simple meaning. Chapter 2 will be for literature review for the past work made on this field. Chapter 3 will be for the methodology that we used; and will explain how our system is able to detect plagiarism on electronic files. Chapter 4 is focused on the experiments that we made on our system and how we can get the best result from it. Chapter 5 concludes our work and presents our observation about the system and what we can do to enhance its result.

## 1.7. Conclusion

In this chapter, we summarizes the information that will let you emphasize the meaning of plagiarism in a simple meaning. We are now understand what is plagiarism detection and their types. In addition, we discuss our main research objectives about the need of a free replacement of the currant systems. In the other hand, we discuss the limitation that we faced on the system and our need to pre-processing the file.

In the next chapter, we will talk about the existing solutions and our literature review for the past work on this field.

# Chapter 2

# LITERATURE REVIEW

# 2. Introduction

This chapter summarizes the most relevant literate review about the recent research on the field of plagiarism in a simple meaning. As we see in the past years, many systems have been developed to check the plagiarism on the basic of searching and matching the tokens with other files. In our literature review, we will be reviewing papers that are related to our works and have the top most techniques; and dived them into the following categories [15]:

## 2.1. Intrinsic Plagiarism Detection

There are many ways to detect the text pieces, sentences or even a block of text copied as whole part even without any external knowledge; but because our scope of the research is to use extrinsic methods, we will summarize some of them to herby understanding of the others works like:

### 2.1.1.Stylistic Consistency Analysis

If one author writes the document, we expect the style change function to remain relatively stable without a notable change. Stamatatos, E. [2] presented a method that find the different style inside the document using the n-grams profiles the group of character n-gram (normalized frequencies of a text) associated to the dissimilar style on the originally suggested style for the author identification. These differences will be used with a group of heuristic rules proposed by the system to minimize the value of the irrelevant style changes within a document, and decide automatically if the document is free from plagiarism or not by measuring the standard deviation (S) that are lower than the predefined threshold (t) using the following formula:

$$S_X = \sqrt{\frac{\sum x^2}{n} - \bar{x}^2} \qquad (1)$$

$$Plagiarism\ free\ criterion: S < t \qquad (2)$$

Stamatatos, E. proposed the following methodology:

- Each word in the document transformed to lowercase.
- Remove every character that contains any not acceptable characters (the accepted are only a-z or any lowercase character of foreign languages) from all document.
- Define a sliding window over the text length and compare the text in the window with the whole document and that give us the function that calculate the style changes inside the document. Figure 2.1 illustrate the change in the style change function.

- Use the peaks of the function to detect the plagiarism inside the document. (Compare sliding window to the whole document).

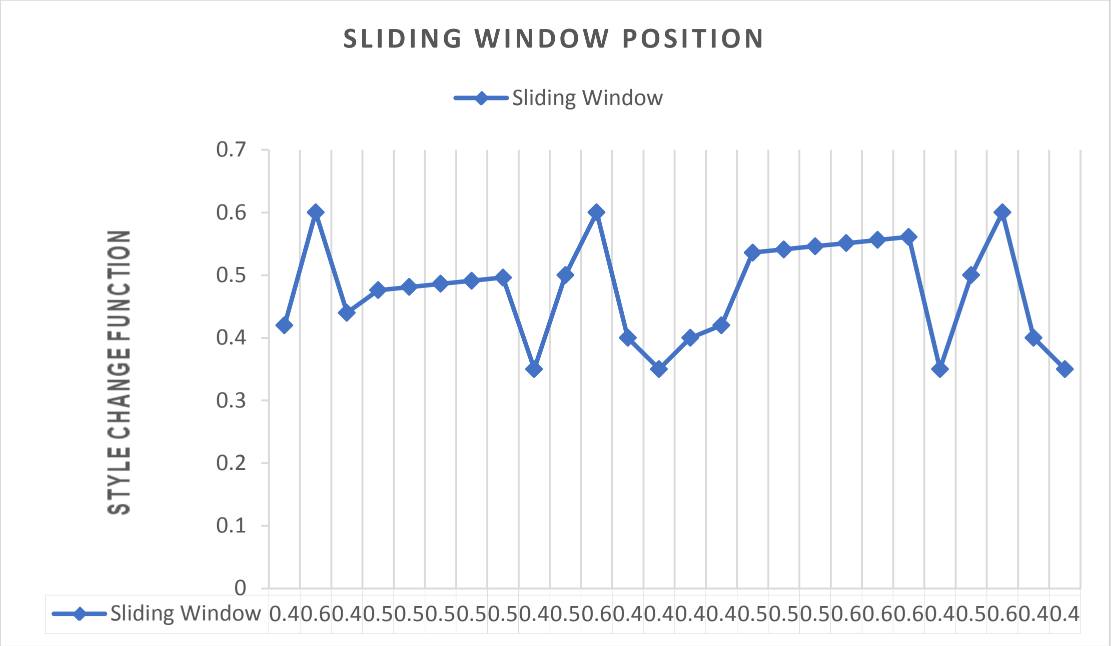


*Figure 2.1: the style change function of the plagiarism-free (a false positive).*

### 2.1.2. Term Occurrence Analysis

However, if we use the style change function to check the file, the speed cost will be high regards to the cost of precision. Zechner, M., et al. [3] presented a method that use a few subjects independent stylometric characteristics from which a vector space model for every sentence of a suspicious document is built. The proposed intrinsic plagiarism detection algorithm is the following:

- Craft a conceptually modest space partitioning method to attain search times in the number of reference documents.
- Calculate the document's mean vector using the following formula:

$$Mean = \frac{1}{N}\sum_{i=1}^{n} x_i \qquad (3)$$

- Build a vector space model for every sentence of a doubtful document.
- Find the outlier sentences based on the document's mean vector.
- Discover plagiarism using outlier analysis which is relative to the document mean vector.
- Assembles the outlier sentences that marked as polarized and made continues blocks of text.

### 2.1.3. Complexity Analysis

Moreover, can we use the machine learning for optimizing results? Seaward, L. and S. Matwin [4] introduce the Complexity Analysis that use Kolmogorov Complexity measures to detect and extract the structural information from document with a small amount of text to be analyzed, this extraction is so important for Intrinsic Plagiarism Detection and can detect if the document is plagiarized or not. They proposed this solution because we can view any sentences as a binary representation. Suppose we represent the noun with 1 and non-noun with 0, then we can construct the binary representation for each word and sentence in the text. We can use this in the calculation since any two sentences might have very similar sense for a specific feature but the distribution can be dissimilar on every one. The proposed algorithm for complexity is the following:

- Segment each of the text and build the distribution *X* related to the word categories. i.e. a 1 for every noun word and a 0 for every non-noun word.
- Use an algorithm to compress the string and this represented by *C(X)*. i.e. The segment *A* will be compressed and transformed to *B*, which has shorter text and can be back by decompression to *A* again.
- That will be used for describing the complexity or degree of randomness of the segment.
- Calculate the Kolmogorov complexity of the binary string using the following formula:

$$Kc(x) = \frac{Length\ C(x)}{Length\ (x)} + q \qquad (4)$$

- Determine if the document is plagiarized or not by checking each passage is more than our selected threshold.

## 2.2. Extrinsic Plagiarism Detection

When we have references of the files that the author maybe plagiarized from, the process of the plagiarism will be more helpful. Many of researchers have developed a set of tools used in external textual automatic detection like:

### 2.2.1. Syntactic Analysis

Alzahrani, S. and N. Salim [5] provided an approach that is using Fuzzy Semantic Based tokenization of the string similarity and search in a list of suspected documents and find their similarity. This approach made by calculating the computation of fuzzy degree of similarity between two sentences i.e. 0 for different sentences and 1 for identical sentences and others are ranged from 0..1.

The proposed algorithm for syntactic analysis is the following:

- Pre-processing that includes tokenization, stemming and stop words removing from the document.
- Retrieving a list of suspicions documents for each document using shingling and Jaccard coefficient using the following formula:

$$Jaccard\ (A,B) = \frac{|shingles\ of\ A| \cap |shingles\ of\ B|}{|shingles\ of\ A| \cup |shingles\ of\ B|} \qquad (5)$$

- Comparing sentence by sentence with the associated candidate documents i.e. they are marked plagiarized if they gain a fuzzy similar above a certain threshold. Figure 2.2 illustrated the accepted threshold of the fuzzy similar.
- Rejoining consecutive sentences to form single paragraphs/sections of text that is plagiarized.

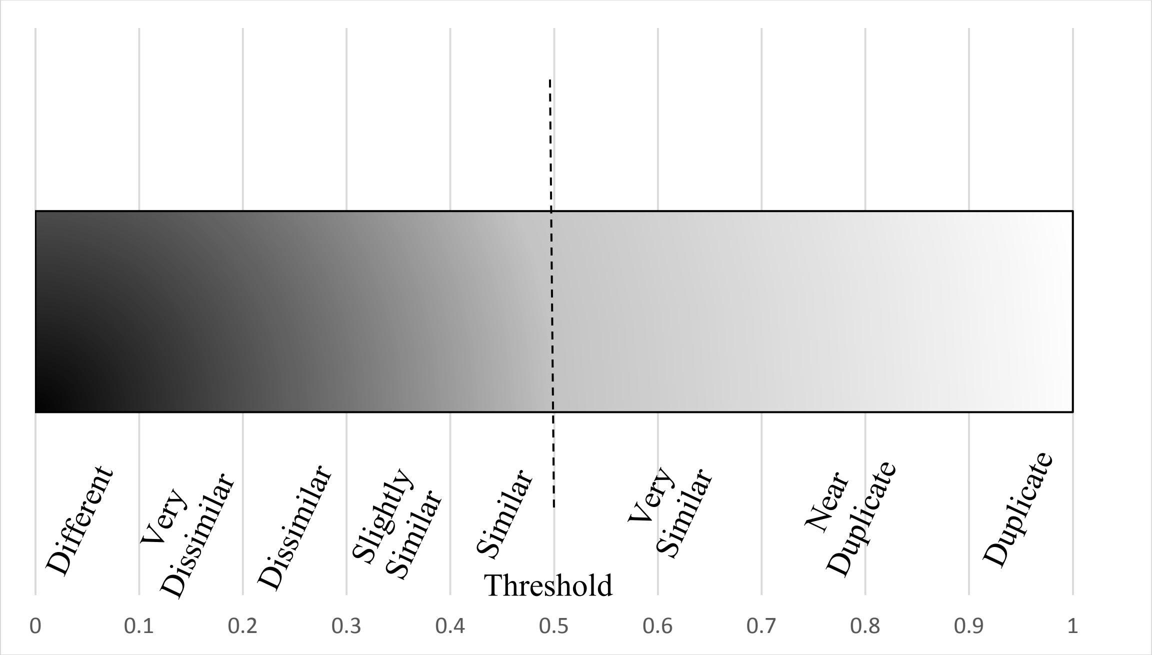


*Figure 2.2 : fuzzy degree of similarity*

### 2.2.2. Word N-Grammar Based Analysis

Ferrero, J., et al. [6] deeply investigate the different methods of Cross-Language Plagiarism Detection Methods and stated that if a method is efficient for a specific language, then it will be similarly efficient on any other language as long as enough lexical resources added for these languages. That has lead us to study Mcnamee, P. and J. Mayfield [7] that introduced a solution by using cross-language semantic textual similarity detection which using the Cross-Language Character N-Gram–just for European language text retrieval-. This typically done by configuring the document and tokenizing the sentences which break words at spaces, downcast them and remove diacritics to identify sentence boundaries to improve accuracy.

The proposed algorithm for text retrieval and analysis is the following:

- Break words at spaces, downcast them and remove any diacritics.
- Identify sentence boundaries by punctuation and then removed.
- Remove English stop phrases from queries –phrase saved and updated over time-. In addition, they are able to be removing from any quires.
- Remove any non-English word to improve accuracy.

- Translated the sentences into the other supported languages using various machine translation systems.
- Comparison with n-grams, each subsequence of length *n* will generated as an n-gram; any text with less *n-2* characters are ignored in the n-grams i.e. they choose 3-grams; sequence of 3 characters.
- Transformed into term frequency–inverse document frequency (TFIDF) vectors [16] of character 3-grams.
- Calculating the similarity between two sentences by metric and compare two vectors is the cosine similarity.

### 2.2.3. Translating Based Analysis

What about taking a document and do a native translation and republish the document? For this problem Pataki, M. [8] introduced a new way for detecting this situation of plagiarism by using the Cross-Language Conceptual Thesaurus-based that measures the distance between sentences and the possible translation of each word in them, and evaluated each sentence possible translation. The author introduced a solution especially between Hungarian and English documents. The developed algorithm was based on the following:

- Search space reduction by removing any stop words and their translating from text.
- Get the language independent form of the text, which we can compare.
- Calculate the distance function between sentences.
- Evaluate document in multiple with a fast candidate search and a precise comparison between possible translations and there distance.
- Define thresholds of similarity: *SimX (Sx)* and *SimY* (Sy) where *Sx* < *Sy*.
- Choose *dMax* and *lMin* that represent the maximum distance and the minimum length of the words in the sentence.
- Calculate the similarity between sentences by calculating the number of common words in different languages using the following formula:

$$Sim\ (Sx, Sy) = min(\alpha.|Sx \cap Sy| - \beta.\left|\frac{Sx}{Sy}\right|, \alpha.|Sy \cap Sx| - \beta.\left|\frac{Sy}{Sx}\right|) \qquad (6)$$

- Selecting the value for $\alpha$ to be 2 and $\beta$ to be 1, meaning matching words are adding 2 points while not matching words are subtracting 1.
- Calculate the document overall similarity metric by joining the sum of all Sim on the sentences.
- Order Documents by their *SIM* values.

In the same way, Barrón-Cedeno, A., et al. [9] used the Cross-Language Alignment-based Similarity Analysis with the help of statistical models that are aims to find the similarity between sentences and the translation that are found in bilingual unigram dictionary [10] which contains translations pairs -and their probabilities- that are already generated by using high performance computers. The problem of this approach is the order of the words are not important, but this assumption is not realistic; there is a huge different of the meaning and cannot be called plagiarism for this matching.

### 2.2.4. Cross-Language Explicit Semantic Analysis

What if we can use a huge dataset like Wikipedia to solve the problem of defining the dictionary? Gabrilovich, E. and S. Markovitch [11] introduced the ability to use the Cross-Language Explicit Semantic Analysis with a high-dimensional space of concepts based on the translation derived directly from the Wikipedia articles - which were defined by humans themselves- compare to the interpretation vectors that are weighted vectors from the original text. This will allow using the new data that will be added over time without worrying of the storage needed and that are available in dozens of languages.

- Fragment the text into pieces; plain text like Wikipedia articles.
- Represent by using the TFIDF [16] vector scheme.
- Get the Wikipedia concepts and sort them by relevance to the text piece by using conventional text classification algorithms.
- Iterate over the text words by using the semantic interpreter that can use Wikipedia concepts directly, without any need for deep language understanding or pre-cataloged common-sense knowledge.
- Get the similarity by using the inverted index.
- Group connected similar pieces into weighted vector of concept.
- Compute semantic relatedness by using cosine metric.

## 2.3. Other Systems Plagiarism Analysis

There is an increasing request of using this knowledge in a working program over time, for the researcher and the institution that will publish any new paper. Further, we will discuss some of existing programs that are published in the field of plagiarism.

Kang et al. [17] present PPChecker for plagiarism pattern checking system that are used to identify and produce more precise result in extracting copy detection with changing for synonyms. Their main comparison is sentences, which can be good in detecting plagiarism on sentences, the paragraph or the whole documents. The architecture of PPChecke contains main three components:

- The Query document component that detect the sentences and prepare them for search.
- The plagiarism unit that search and find similarity on the local and inside the document.
- The local document Database that will be used for any future search on this program.

Meanwhile, Jiffriya et al. [18] presented AntiPlag, another way for detection using optimizing and enhancement through the clustering. This enhanced made the AntiPlag fast and capable of comparing all plagiarized pairs of sentences automatically at once.

On the other hand, the field of the Arabic Plagiarism are rising in the same way; Bensalem et al. [19] presented plagiarism on Arabic textual documents with Stylysis tool using a group of initial experiments on intrinsic plagiarism discovery in Arabic text and discovered that vocabulary is the main problem in Arabic plagiarism.

Furthermore, Menai, M.E.B. [20] presented APlag that are capable to detect the sentence structure change and synonym replacement on Arabic documents. The architecture of APlag contains main four components:

- Preprocessing the document: tokenize the text, remove any stop-word and replace synonym.
- Fingerprinting: by using of n-grams, where *n* is chosen by the user.
- Document representation: represent the internal structure of the document by using the tree algorithms for each document.
- Similarity metric: find the longest match of the two hashed strings

Alternatively, Turnitin [21] a highly famous detection tool is capable to search for plagiarism and used for detecting text coping over their own database of papers [15]. Dahl, S. [22] Published an exploratory study that examines how students use the Turnitin and what are their feedback about such system. The majority of the students in this study were mostly optimistic about Turnitin. Some of the student sample favor to use electronic program instead of the old way to give it to the designated office for checking, and are positively want to decrease the plagiarism ratio in their submissions. Dahl, S. found that the student dived into two categories; one is know how to make a quote correctly and are happy to check with such programs to avoid plagiarism. In the other hand, the other students who are less happy for such program because of their limitation of quote correctly that meaning of plagiarism. As a result, introducing such programs and make a student use them easily will help and have a major change in the view of the students.

## 2.4. Conclusion

In this chapter, we review the most relevant literate about the recent research on the field of plagiarism in a simple meaning. We are now understand what is intrinsic plagiarism detection and their types. In addition, we discuss extrinsic plagiarism detection and their types. In the other hand, we analyze and discuss the current systems that are in the same field.

In the next chapter, we will talk about our methodology; and how we will detect the plagiarism in the electronic files.

# Chapter 3

# RESEARCH METHODOLOGY

# 3. Introduction

This chapter summarizes the methodology that used to detect the plagiarism on the electronic files in an intelligent way. Artificial intelligence means creating software that emulates the characteristics of human intelligent [42-50]. Therefore, we can discuss the methodology in clear terms we need to talk about the following:

## 3.1. Tools Help in the Methodology

Before we talk about our methodology in detail, we need to talk about some tools that made this work possible is:

### 3.1.1. Delphi

Delphi [23] is an IDE that help to build programs with fast and easy way. This program was selected for its feature like cross platform native application that can generate from the same source. The main programming language is the Modern Object Pascal language. On the same hand, the high resources in the component that written and founded easily in Delphi and we can use it very easily.

We select Delphi 2010 for the purpose of development because it is the main language we use in some of the university programs and this will help to adapt the program later whenever any new update are found.

### 3.1.2. MS Office DLL

Microsoft Office [24] is suite of programs that Microsoft present to help for different purpose like Word, Excel and PowerPoint.

We will use the provided DLL in the windows that Microsoft provide using the OLE (Object Linking and Embedding) [25] that will open a word or pdf document and extract the source of the file and get the text only to help us in the process of plagiarism detection.

### 3.1.3. DuckDuckGo

DuckDuckGo [14] is a search engine which doesn't keep track of you on the Internet is a search engine that is concerned about the user privacy in searches and provide results from a variety of 100 sources like: Wikipedia [26], Wikia [27], CrunchBase [28], GitHub [29], WikiHow [30], The Free Dictionary [31] – over 100 in total [32]. This made this site rank and use go higher every day.

We select DuckDuckGo because they provide API that serves over 10,000,000 queries per day. In addition, we can use deferent customization inside the search process like SITE , quoting and so more [33].

### 3.1.4. HTML

HTML is the regular markup language for generating Web pages [34]. This language describe the web page that one access all the times on the internet.

#### 3.1.4.1. DIHtmlParser

DIHtmlParser [35] is a component suite that developed for Delphi that can parse, analyze, extract information from, and generate HTML, XHTML, and XML documents from web.

We select DIHtmlParser because it provides a full Unicode support that meaning support for any language and for the capability of extended easily by using TDIHtmlParserPlugin interface.

#### 3.1.4.2. DIDuckDuckGoReader

DIDuckDuckGoReader is our developed version of the TDIHtmlParserPlugin that DIHtmlParser provide. With this customized reader, we can easily give them the HTML document and they parse it.

#### 3.1.4.3. SuperObject

A fast Delphi JSON (JavaScript Object Notation) [36] parser that provided on the GitHub [37] and can parse JSON easily for the result of DuckDuckGo API requested.

### 3.1.5. Regex ~ Regular Expression

Regex [38] is an expression that we can build from finite sets of strings using the operations of union, concatenation, and Kleene star. For example, see the text below:

p:444-555-1234 f:246.555.8888 m:1235554567

After using this regex, it can be used to detect number in the string like the following formula:

$$/\backslash d\langle 3\rangle[-.]?\backslash d\langle 3\rangle[-.]?\backslash d\langle 4\rangle/ \qquad (7)$$

The result will be:

p:444-555-1234 f:246.555.8888 m:1235554567

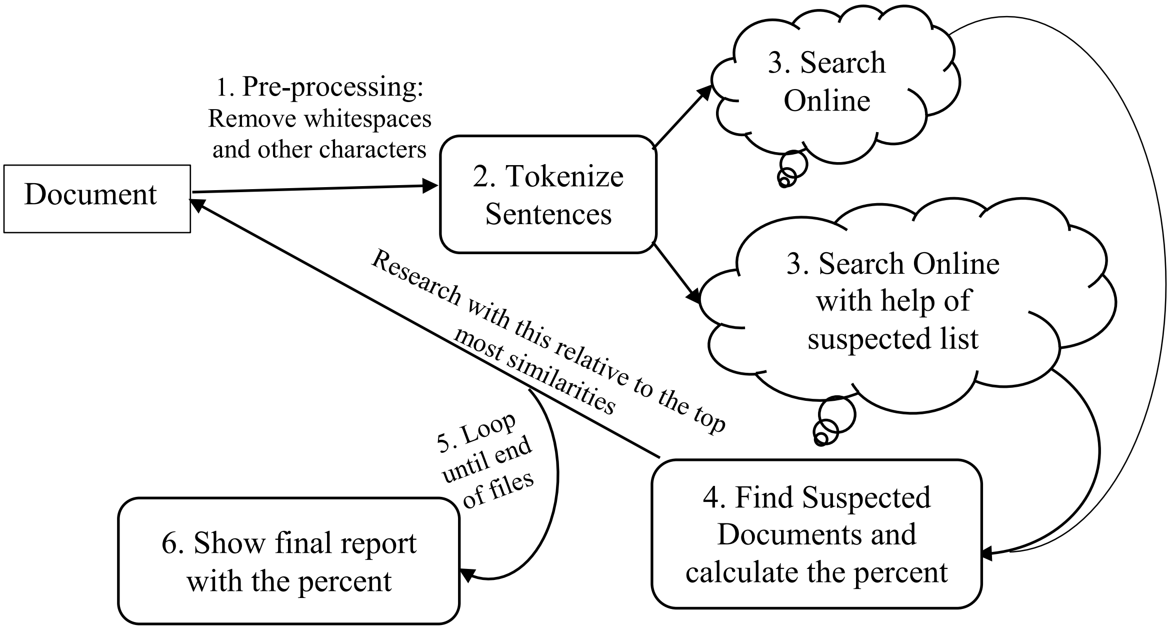
The part $\backslash d\langle 3\rangle$ describe that should select 3 decimal. Followed by $[-.]$? meaning they can have connective '-' or '.' or nothing. Then another 3 decimal. Then '-' or '.' or nothing. Then 4 decimal. Yes regex looks that Easy!

This expressing mainly used usually with string searching algorithms to find or replace operations on set of strings.

### 3.1.5.1. PerlRegEx

PerlRegEx [39] is a set of free to use classes that build for use Regex in Delphi. This library is perform the regex searching algorithm in the given text.

## 3.2. Proposed Methodology



1. Pre-processing:
Remove whitespaces
and other characters
Document
2. Tokenize Sentences
3. Search Online
3. Search Online with help of suspected list
Research with this relative to the top most similarities
5. Loop until end of files
4. Find Suspected Documents and calculate the percent
6. Show final report with the percent

*Figure 3.1: the Overall Methodology of Proposed Framework*

Figure 3.1 illustrate our modified framework that will consist of the following:

1. Pre-processing the file and remove any Punctuation marks and Diacritics from the text and remove any special character like character formation in Arabic Language.
   - This will be using the help of mathematical Regex:
     - ❖ The first regex will be the main Arabic and English letters with Numeric characters. That will help remove any others letter.
     - ❖ The second regex will be the main Arabic Diacritics and remove them from the words.

Furthermore, Read text word by word, this will be using the help of mathematical Regex:

- The regex will divide the string word by word to make a token of the string by detecting every space in the document.

2. Tokenize the words based on a fixed sliding window of text that can be changing by the user.
   - We select the number of words to be 12 in the token –the count of the words in one line in the document; and the user-can change this in runtime-.

Here we will have 2 types of token; one that is cleared from any unwanted text, the other will be the text as they written without any modification – This will help in the quoted search-.

3. Search for the token-sliding window- over the web and got the result.

The selected mechanism for the search any token is the following:

- Start search string by quoting "" the token; this will be helpful for finding the exact match of the string -here we will use the type of token that is not altered by any way-.
    - If a match is found add the source of the file to a constructed list of site that will be helpful for gathering the suspected documents.
- Second search token without quoting, this will be helpful for finding the semi match of the string with the help of the rules of the search engine that are searching for any part or synonym of the token -here we will use the type of token that altered by our system-.
    - If a match is found add the source of the file to a constructed list of site that will be helpful for gathering the suspected documents.
- Third with the help of the constructed list of suspected document; search for the token with specific search in that source thanks for the rules of the search engine that can specified in a site by adding SITE: to the query -here we will use the type of token that altered by our system to get the best match-.
- We will add a specific link in the top of the list that will be our university site. This will help the system in searching all university local documents without the need to search and save documents in our system; meaning no need for any extra storage for the search.
    - ❖ The default defined maxed search for any token will be 3 general searches, with adding specific search with the size of the suspected document list.
    - ❖ Optional: The user can add suspected source to the list manually and the system will search for the list that will be ordering by the frequency of the plagiarism that found in it per search.

4. Download the result and extract the exact result for the search and calculate the token plagiarism percentage.
    - Parse the result of the search and get the top ranked searches in the result.
    - As same as the pre- processing we will remove any text ~ return to point 1, 3.
    - Divide the result to 3 block of text and get every probability of the connecting string to search in it; this will be helpful to get the approximate percent of matching.
    - The same we divide the result as we divided the tokens.
    - Now by loping to both list found the matching token percent by using the

following formula:

$$Plagirism\ Perecent = Floor\left[\left(\frac{Length\ Of\ Founded\ Matching\ Array}{Length\ Of\ Orginal\ Array}\right) * 100\right]\% \qquad (8)$$

- ❖ The default defined maximum token plagiarism threshold selected as 75% that mean we will mark the full token 100% plagiarism if the percent > 75% .
- ❖ We can have the percent more than 100% because of the probability of having more combination valid in the string and this will be down to 100%.

5. Loop throw tokens until final token as same before until the end of file and calculate the major token plagiarism percentage for the whole file using the following formula:

$$Document\ Plagirism\ Perecent = \frac{\sum Plagirism\ Perecent}{Number\ of\ tokens\ per\ document} \qquad (9)$$

- ❖ We have selected the maximum percentage to be 25% of the total tokens in the file to be marked as plagiarized document.

6. Prepare the report with feedback needed to the researcher or university assistance.

## 3.3. Conclusion

In this chapter, we summarizes the methodology that we used to detect the plagiarism on the electronic files in an intelligent way. In addition, we discuss the tools that help in the detection process and how we are using each one in a simple meaning.

In the next chapter, we will talk about our experiments on the methodology and discuss our result that we got from the system.

# Chapter 4

# EXPERIMENTS

# 4. Introduction

This chapter summarizes the experiments that we tested the methodology and check the plagiarism in the electronic file.

## 4.1. Dataset

After we present our methodology, we want to test the system; in mean of the best percentage of detection can catch or based on the performance evaluation that we need to measure to satisfy the best cases for our system. However, for making this happen, we need some real plagiarism situation to test in our system; unfortunately, this cannot be afford because we need large numbers of documents to make our test accepted. On the other hand, we cannot use any real documents that had distributed without having the permission of the owner and we cannot use any generated or free text documents with respect to our honesty point of view. This led us to make a simulated plagiarism situation that helps us in testing the performance and the acceptance of the methodology.

For testing purposes, our selected dataset consist of two types: first we will use about 100 different corpus consisting of short (200-300 words ~ English words) that Clough, P. and Stevenson, M. [40] developed in which plagiarism has been simulated. The other type will be checking over different topics like scientific, engineering literature, general news and static pages from the web in both languages: English and Arabic languages.

The plagiarized corpus consists of five learning task that illustrated on Table 4.1 and consist of the following types:

A. What is inheritance in object oriented programming?
B. Explain the PageRank algorithm that used by the Google search engine.
C. Explain the Vector Space Model that is used for Information Retrieval
D. Explain Bayes Theorem for probability theory.
E. What is dynamic programming?

The generated corpus plagiarism ranged from 19 file that are near copy (100%..75%), 19 file that are light revision (75%..50%), 19 file that are Heavy revision (50%..25%) and 38 file that are Non-plagiarism (25%..0%). The total of 95 file that will used and this will helps us to calculate the accuracy of our system in respect to this average percent.

*Table 4.1: numbers of tested corpus and their categories*

| ***Category*** | ***Learning task*** | | | | | ***Total*** |
|---|---|---|---|---|---|---|
| | **A** | **B** | **C** | **D** | **E** | |
| ***Near Copy*** | 4 | 3 | 3 | 4 | 5 | **19** |
| ***Light Revision*** | 3 | 3 | 4 | 5 | 3 | **19** |
| ***Heavy Revision*** | 3 | 4 | 5 | 4 | 3 | **19** |
| ***Non-Plagiarism*** | 9 | 9 | 7 | 6 | 7 | **38** |
| ***Total*** | **19** | **19** | **19** | **19** | **19** | **95** |

## 4.2. Performance Evaluation

What about the performance, how we can test that? Potthast, M., et al. [41] develops a framework that provides many performance measures and address the performance of plagiarism detection systems. They introduce 3 measures that we can apply one by one; or in combined with each other. In order to test our system, we need to describe some important parameters that Potthast, M., et al. introduce. Let $\boldsymbol{S}$ be the source document, let $\boldsymbol{R}$ be the plagiarism detection for the document,

$$\boldsymbol{Dplg}:\ denote\ the\ document\ that\ contain\ plagarism,$$

$$\boldsymbol{S} = \langle Splg\ , Dplg\ , Ssrc\ , Dsrc\rangle, where\ Splg\ is\ palagirized\ passage\ in\ Dplg,$$
$$and\ Ssrc\ is\ the\ orginal\ part\ from\ the\ document\ Dsrc$$

$$\boldsymbol{R} = \langle Rplg\ , Dplg\ , Ssrc\ , D'src\rangle, where\ Rplg\ is\ palagirized\ passage\ in\ Dplg,$$
$$and\ Rsrc\ is\ passage\ from\ the\ document\ D'src$$

We say that $\boldsymbol{R}$ detect the document,

$$\boldsymbol{Iff}\ Rplg \cap Splg\ \neq 0, Rsrc\ \cap Ssrc\ \neq 0, and\ D'src = Dsrc$$

We will use the following tests to check our performance.

### 4.2.1. Test 1 : Precision

Precision (positive predictive value): defined as the test for the closeness of two or more values to each other. We can use precision to measure the performance of our system as using the following formula:

$$\boldsymbol{Prec}(\boldsymbol{S},\boldsymbol{R}) = \frac{1}{|R|}\sum_{r\in R}\frac{|\bigcup_{s\in S}(s\sqcap r)|}{|r|}, \qquad (10)$$

$$where\ \boldsymbol{s} \sqcap \boldsymbol{r} = \begin{cases} s \cap r & if\ r\ detects\ s, \\ 0 & otherwise. \end{cases}$$

### 4.2.2. Test 2 : Recall

Recall (sensitivity): defined as the proportion of positives values that have correctly identified by the system. We can use recall to measure the performance of our system as using the following formula:

$$\boldsymbol{Rec}(\boldsymbol{S},\boldsymbol{R}) = \frac{1}{|S|}\sum_{s\in S}\frac{|\cup_{r\in R}(s\sqcap r)|}{|s|}, \qquad (11)$$

$$where\ \boldsymbol{s}\sqcap\boldsymbol{r} = \begin{cases} s\cap r & if\ r\ detects\ s, \\ 0 & otherwise. \end{cases}$$

### 4.2.3. Test 3 : Granularity

Granularity (the level of detail): defined as the scale or level of detail that is present in a set of data. We can use granularity to measure the performance of our system as using the following formula:

$$\boldsymbol{Gran}(\boldsymbol{S},\boldsymbol{R}) = \frac{1}{|S_R|}\sum_{s\in S_R}|R_S|\,, \qquad (12)$$

$$where\ \boldsymbol{S_R} = \{s \mid (s \in S) \wedge (\exists r \in R) : r\ detects\ s\} \sim \text{Prec} + \text{Rec}\,,$$

$$and\ \boldsymbol{R_S} = \{r \mid (r \in R) \wedge r\ detects\ s\} \sim \text{Number of matching has reported}$$

### 4.2.4. Overall Test : Precision, Recall, and Granularity

To obtain the absolute result, we must select the plagiarism detection to be an overall score as using the following formula:

$$\boldsymbol{OverAll\ PlagDetect}(\boldsymbol{S},\boldsymbol{R}) = \frac{F_\alpha}{log_2(1+Gran(S,R))}\ , \qquad (13)$$

$$where\ F_\alpha denotes\ the\ F_\alpha Mesure,\ \boldsymbol{F_\alpha}\ \boldsymbol{Mesure}\ (\boldsymbol{S},\boldsymbol{R}) = \frac{2\ .Prec\ .Rec}{Prec+Rec} \qquad (14)$$

Because there are no indications that which one (Precision or Recall) is more important, the suggestion is to use $\alpha = 1$ (precision and recall equally weighted). On the same hand, the selection of the granularity measure is to decrease its impact on the overall score.

## 4.3. Results and Discussion

After the definition of the terms that we are going to use in the detail comparison, we will compare out system in match to the specified methods above. First, we need to find the best values of the methodology parameters for which the detection results (Precision, Recall, Overall Test) will be the best. These parameters are illustrated in table 4.2 and consist of the following:

*Table 4.2: selected parameters for our methodology*

| ***Name*** | ***Description*** | ***Range*** | ***Selected Value*** |
|---|---|---|---|
| $\alpha$ | Precision and Recall equally weighted as Potthast, M., et al. described. | **1..10** | **1** |
| $\beta$ | Maximum number of words in the token. | **3..28** | **12** |
| $\gamma$ | The maximum search times for any token, where n is the size of the suspected document list. Where +1 is our university site that added on the list. | **1..n** | **3..n+1** |
| $\delta$ | The maximum search connected result that can mark as plagiarized. | **1..$\beta$** | **3..$\beta$** |
| $\varepsilon$ | The maximum token plagiarism threshold for any token to mark as plagiarized. | **1..100%** | **≥ 75%** |
| $\zeta$ | The maximum token plagiarism threshold for the whole document to mark as plagiarized. | **1..100%** | **≥ 25%** |

We are now able to configure our system and select the best value for each parameter. Let us start by selecting the best **β :**

### a) Configuration $\beta$

Figure 4.1 illustrated the selecting of *3* word on our token; therefore we can get unwanted behavior as we see some files are get above 100% of the ratio.

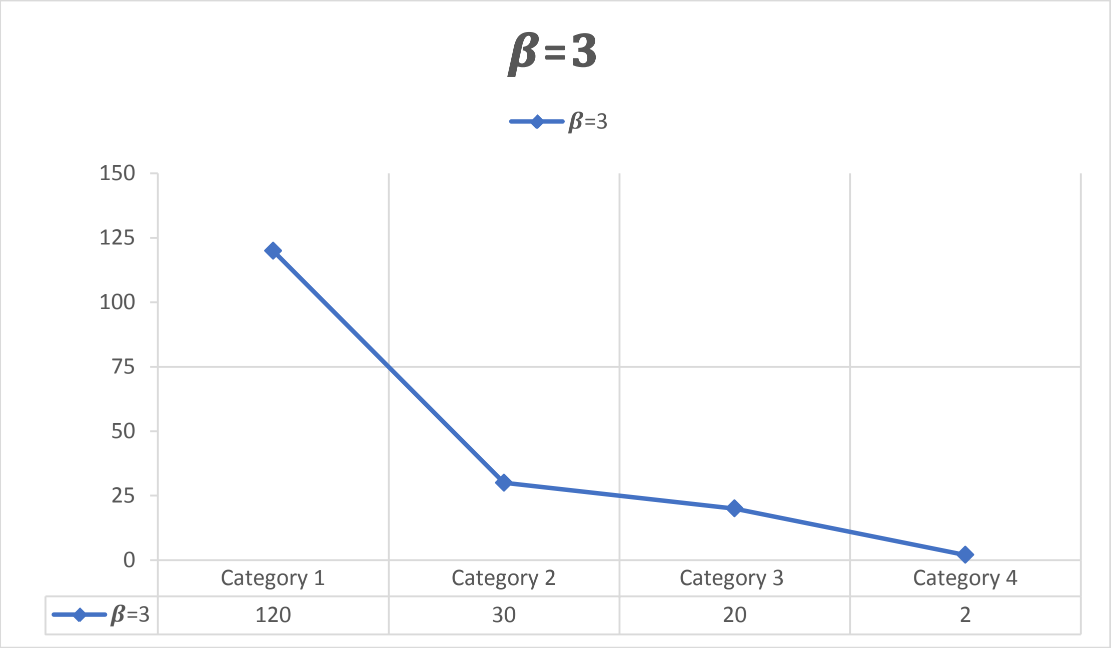


*Figure 4.1: configuration of $\beta$ = 3*

Figure 4.2 illustrated the selecting of *5* word on our token; as we see the percent enhanced but we can't get any percent about category 4.

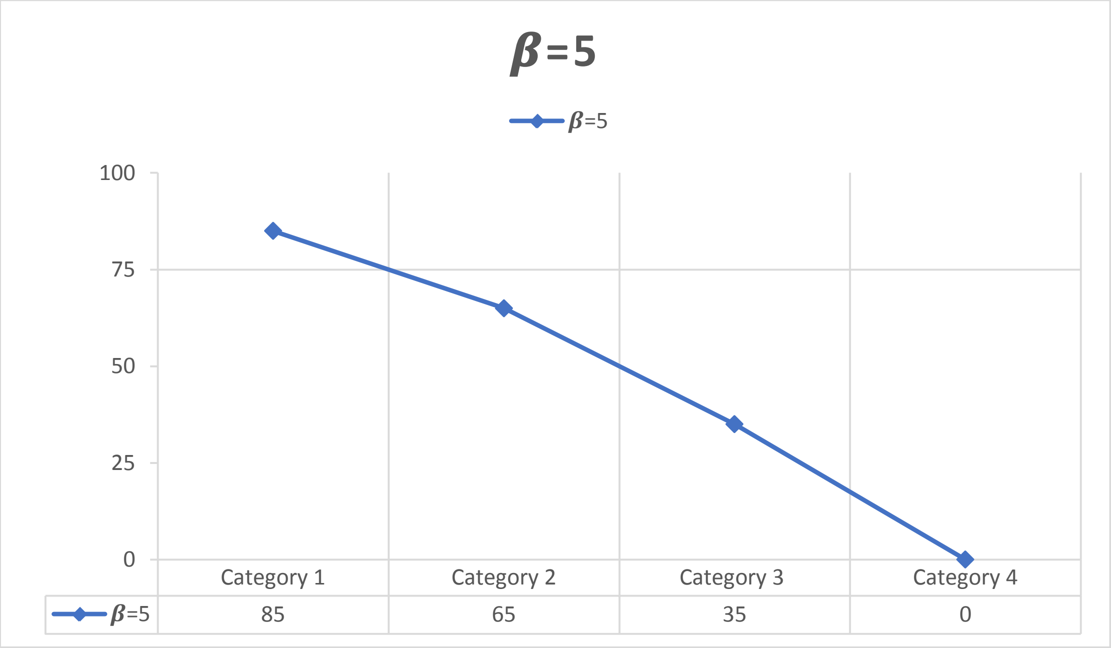


*Figure 4.2 : configuration of* $\boldsymbol{\beta} = 5$

Figure 4.3 illustrated the selecting of *8* word on our token; as we see the percent enhanced in category 4 but dropped in other category.

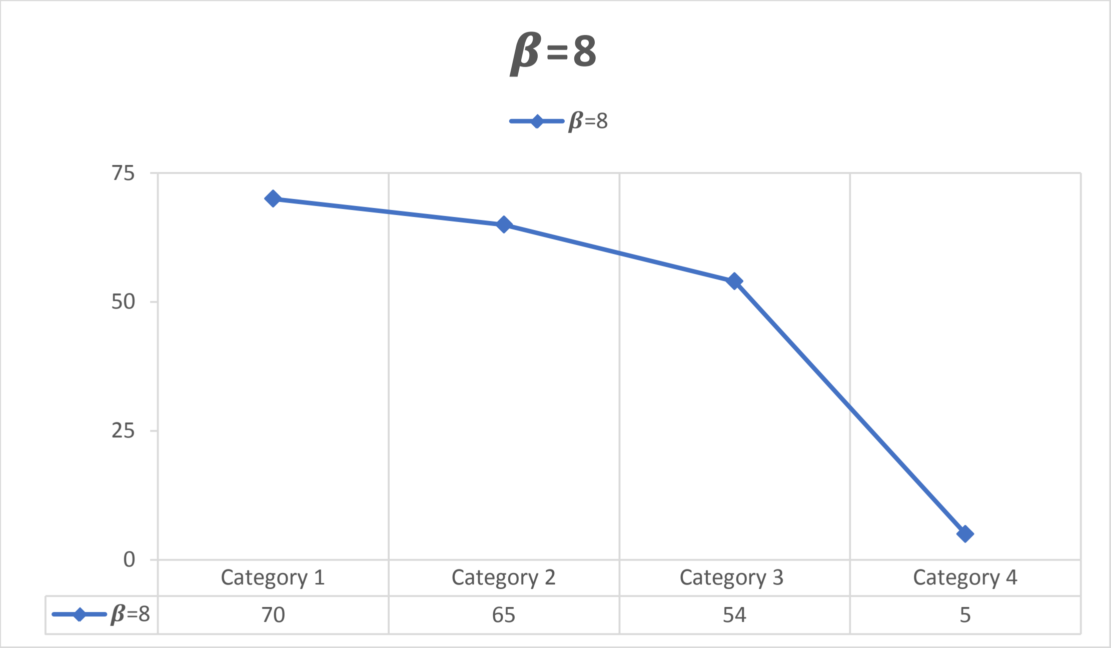


*Figure 4.3 : configuration of* $\boldsymbol{\beta} = 8$

Figure 4.4 illustrated the selecting of *12* word on our token; as we see the percent enhanced in category 4 and still good in other category.

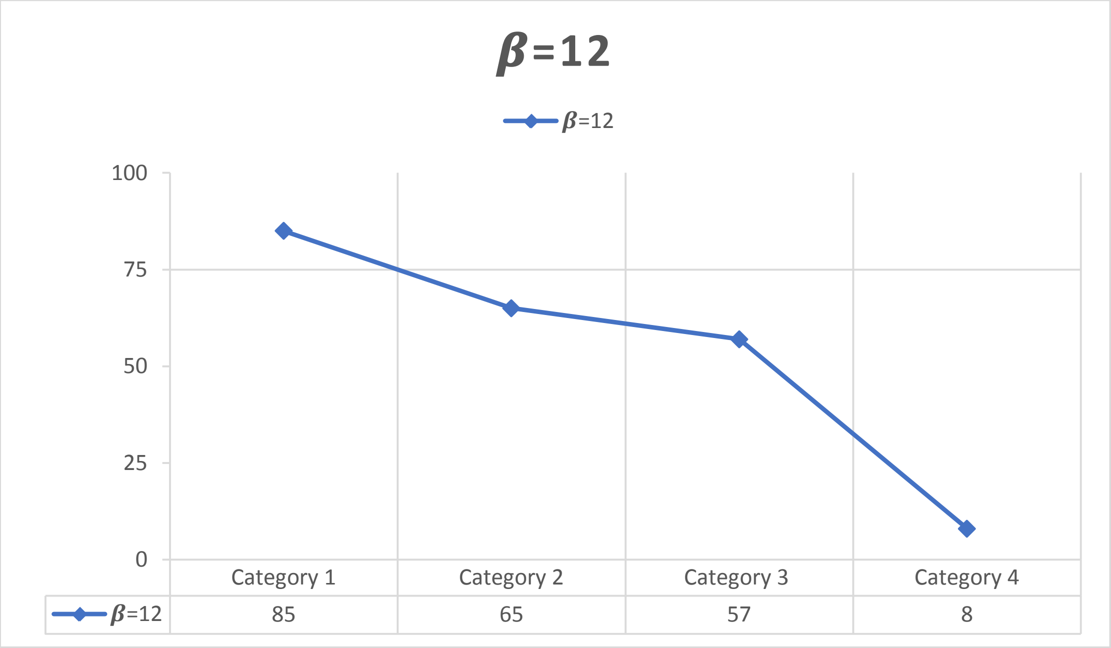


*Figure 4.4: configuration of* ***β*** *= 12*

Figure 4.5 illustrated the selecting of *20* word on our token; as we see the percent start dropping in all category.

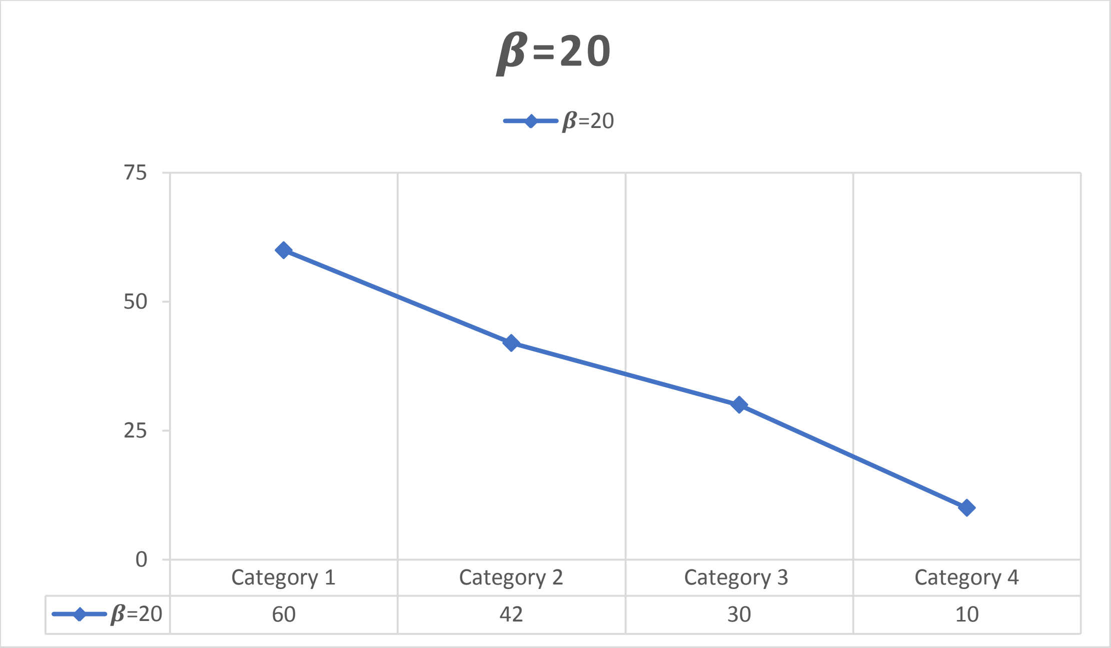


*Figure 4.5: configuration of* ***β*** *= 20*

Figure 4.6 illustrated the selecting of *28* word on our token; as we see the percent continue dropping in all category.

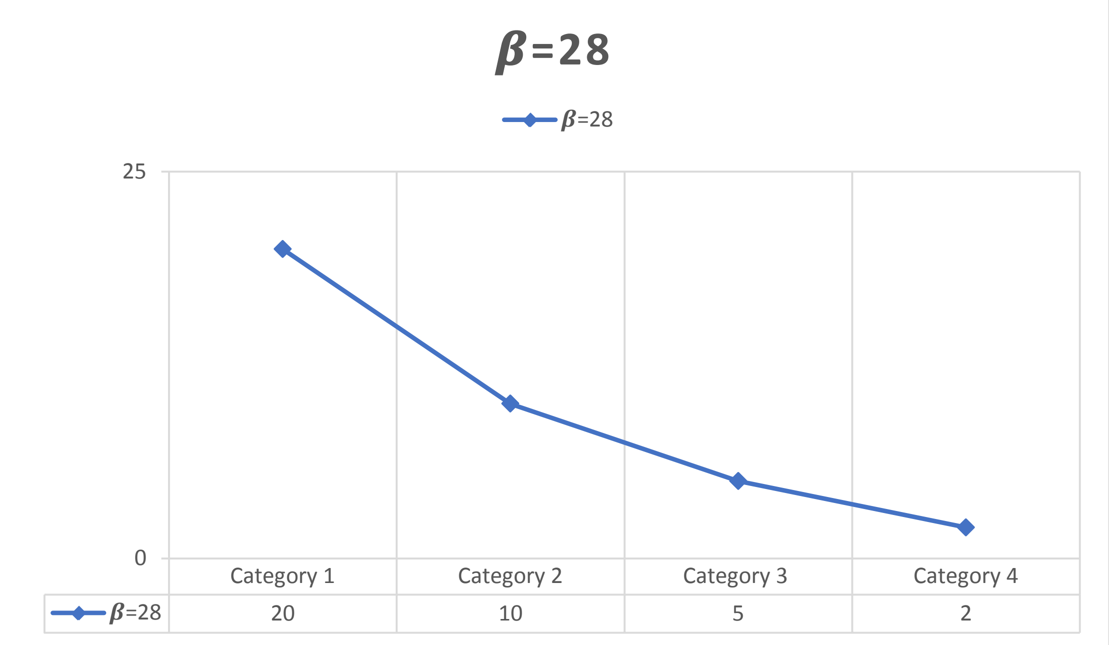


*Figure 4.6 : configuration of* $\boldsymbol{\beta}$ *= 28*

Figure 4.7 illustrated the selecting of *30* word on our token; as we see the system cannot find any match and this because the search engine is ignore the high word in the search token.

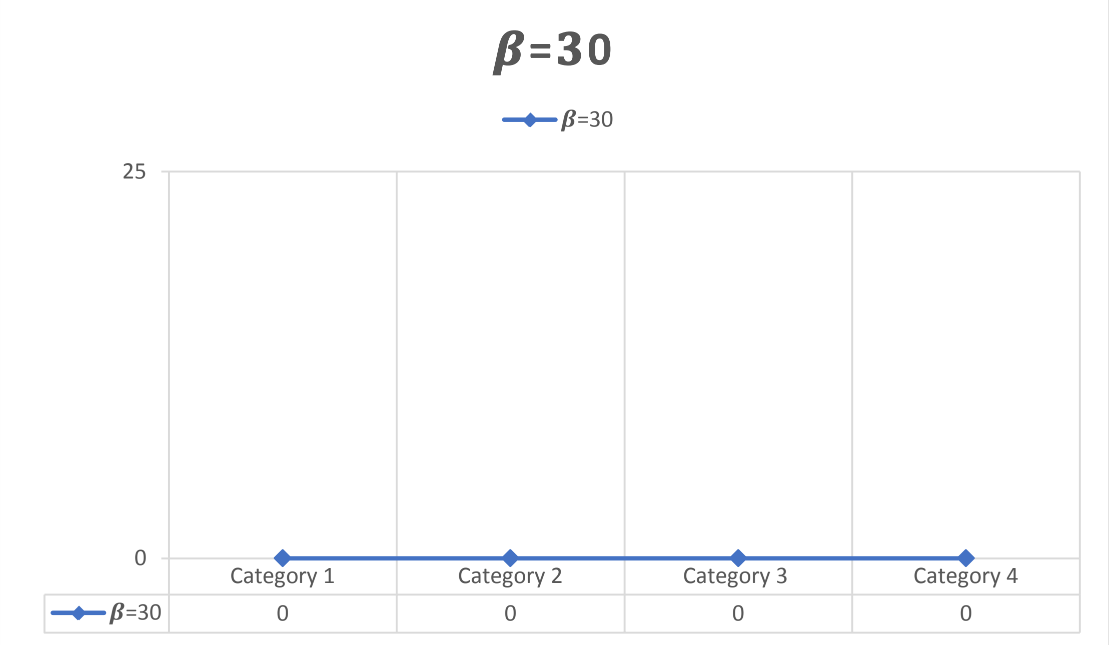


*Figure 4.7 : configuration of* $\boldsymbol{\beta}$ *= 30*

After selecting the best **β** which was *12*, now we will start selecting the best **$\gamma$:**

b) **Configuration $\gamma$**

Figure 4.8 illustrated the selecting of *1* search time for our token; therefore we can start getting values and ratio from the web.

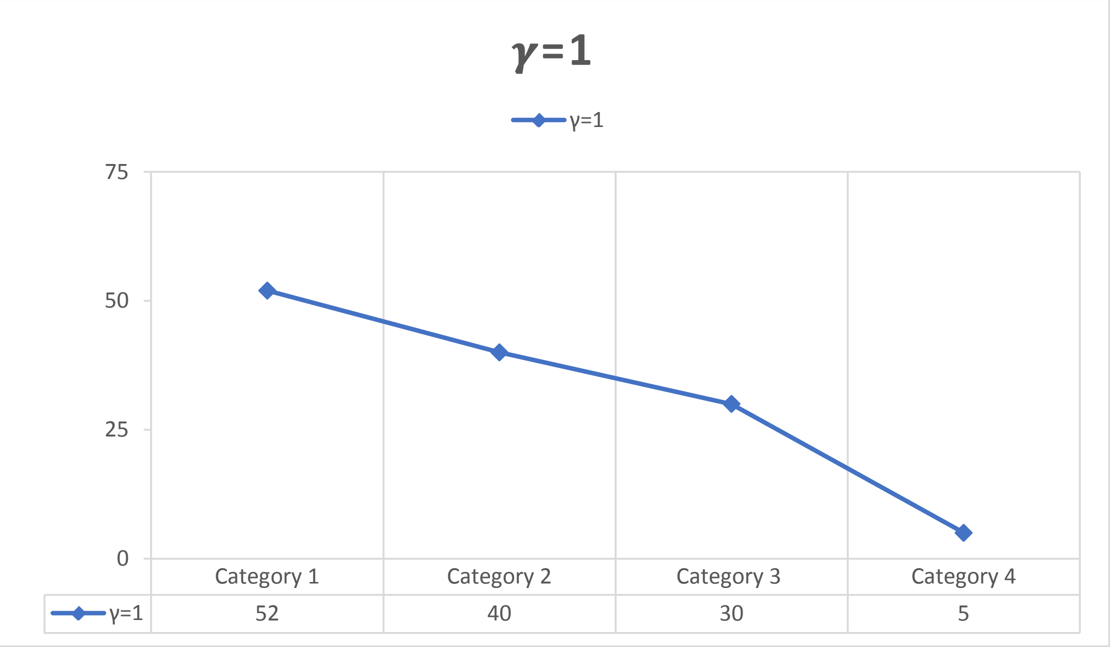


*Figure 4.8 : configuration of γ = 1*

Figure 4.9 illustrated the selecting of *2* search time for our token; therefore we can start enhance our ration from the web.

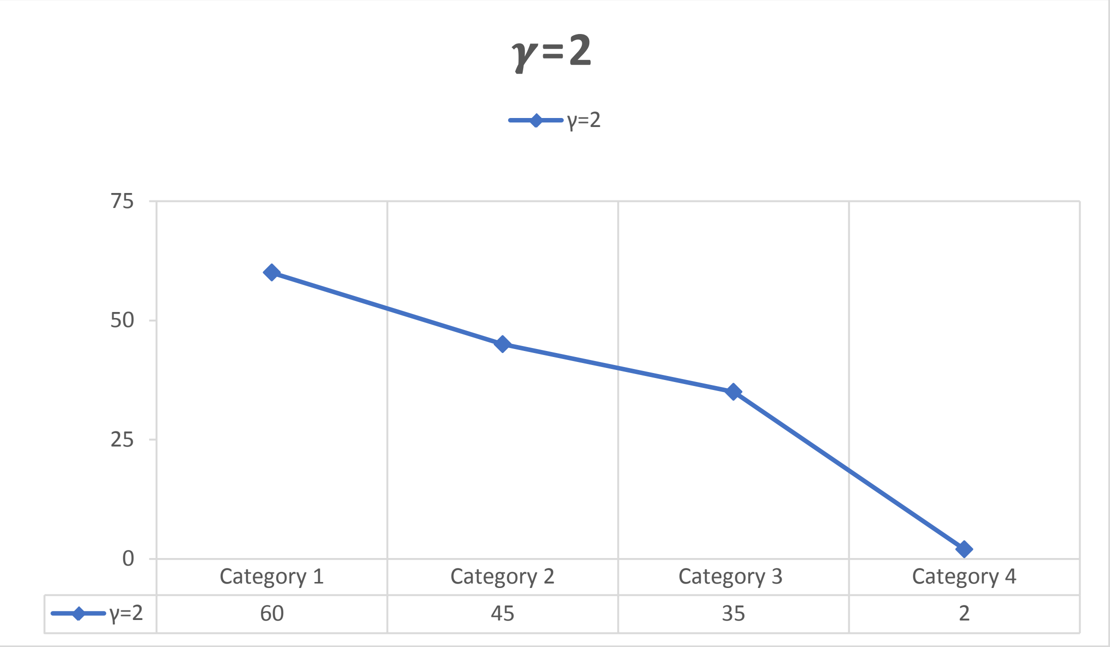


*Figure 4.9 : configuration of γ = 2*

Figure 4.10 illustrated the selecting of *3* search time for our token; therefore we can continue enhance our ration from the web.

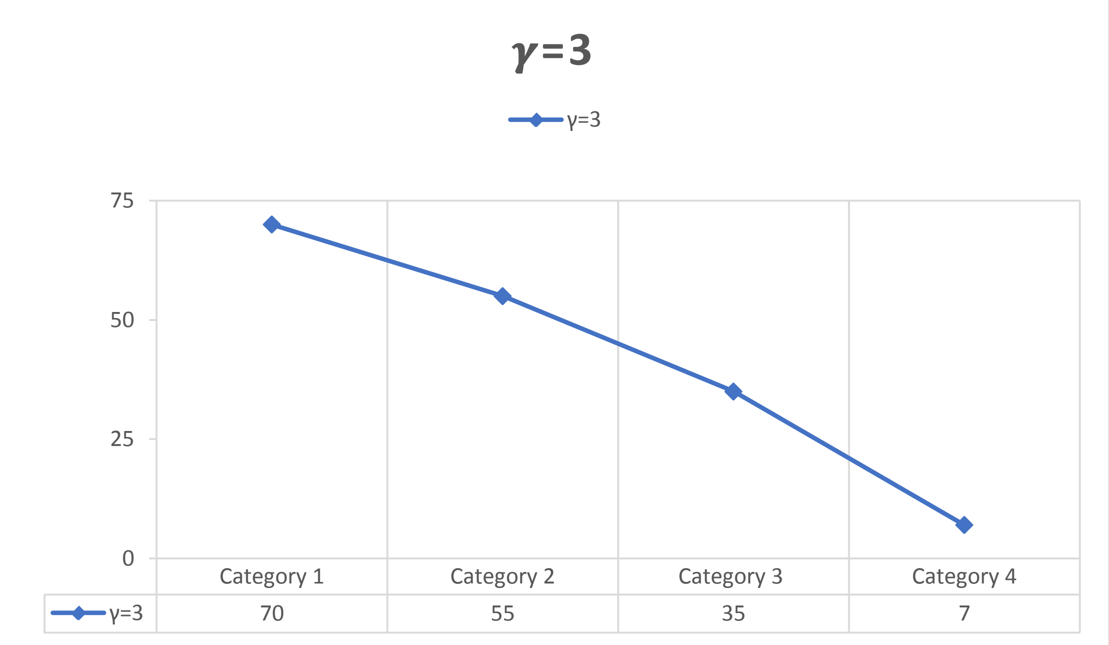


*Figure 4.10 : configuration of γ = 3*

Figure 4.11 illustrated the selecting of *3..N+1* search time for our token; therefore we enhance our ration to be from web and from our suspected list that are growing throw search .

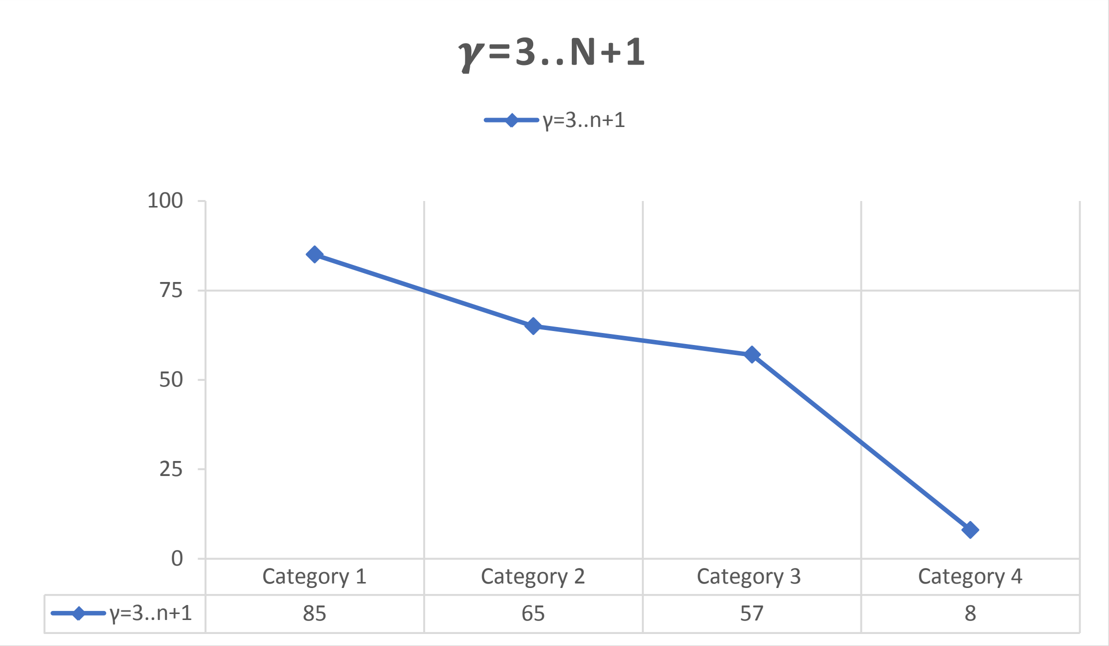


*Figure 4.11 : configuration of γ = 3..n+1*

After selecting the best $\gamma$ which was *3..N+1* search time for our token, now we will start selecting the best **$\delta$:**

c) **Configuration $\delta$**

Figure 4.12 illustrated the selecting of *1* connected word for any token to mark as plagiarized; therefore we can start getting values and ratio from the web.

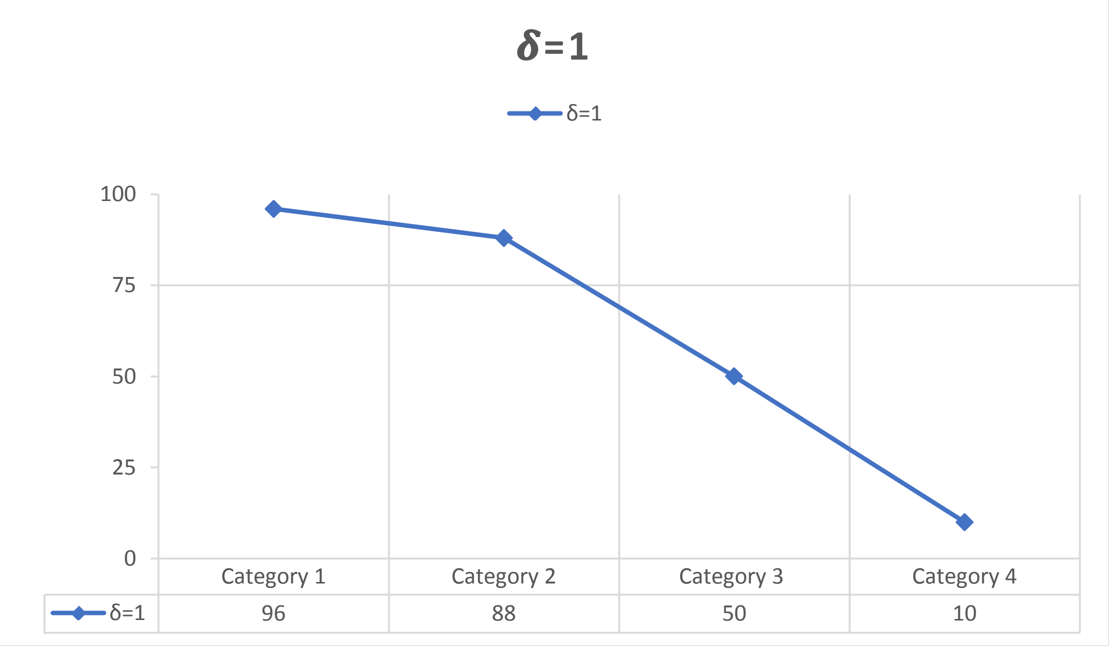


*Figure 4.12 : configuration of* ***δ****=1*

Figure 4.13 illustrated the selecting of *2* connected word for any token to mark as plagiarized; therefore we the ratio.

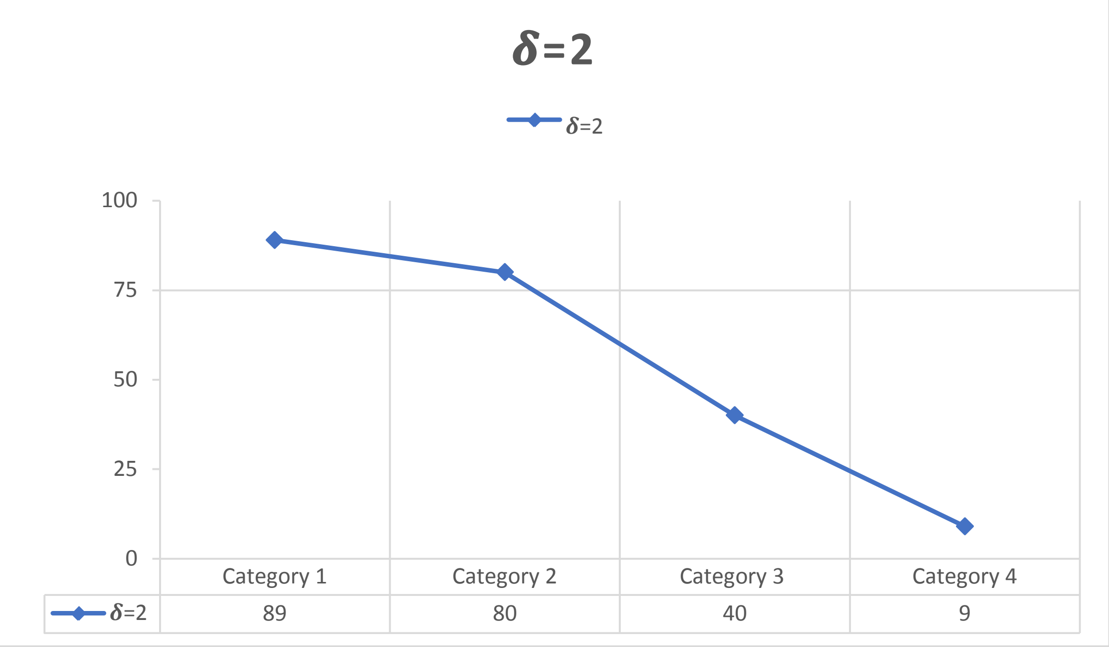


*Figure 4.13 : configuration of* ***δ****=2*

Figure 4.14 illustrated the selecting of *3* connected word for any token to mark as plagiarized; therefore, we enhance the ratio in category 2 and 3.

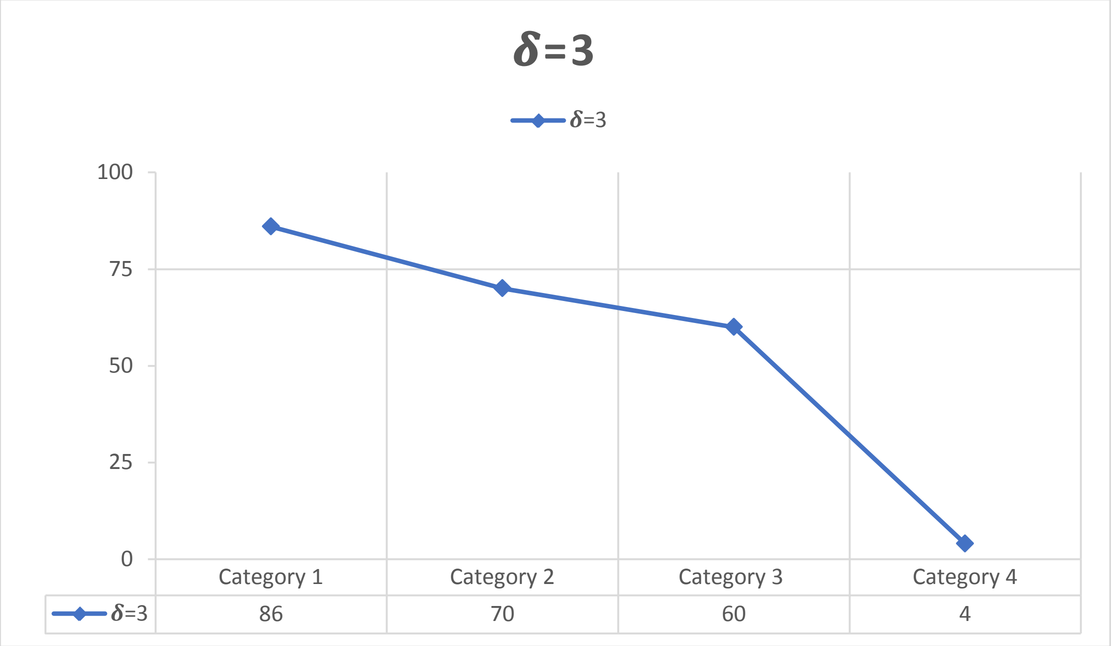


*Figure 4.14 : configuration of* ***δ****=3*

Figure 4.15 illustrated the selecting of *8* connected word for any token to mark as plagiarized; therefore, ratio start dropping.

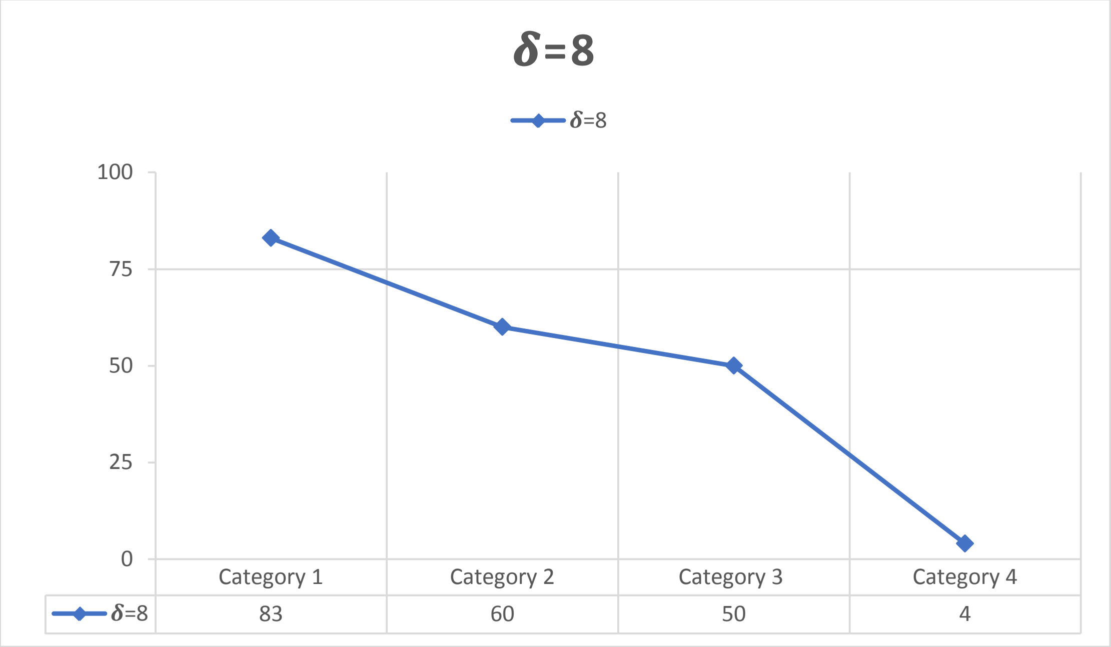


*Figure 4.15 : configuration of* ***δ****=8*

Figure 4.16 illustrated the selecting of *12* connected word for any token to mark as plagiarized; therefore, ratio continue dropping.

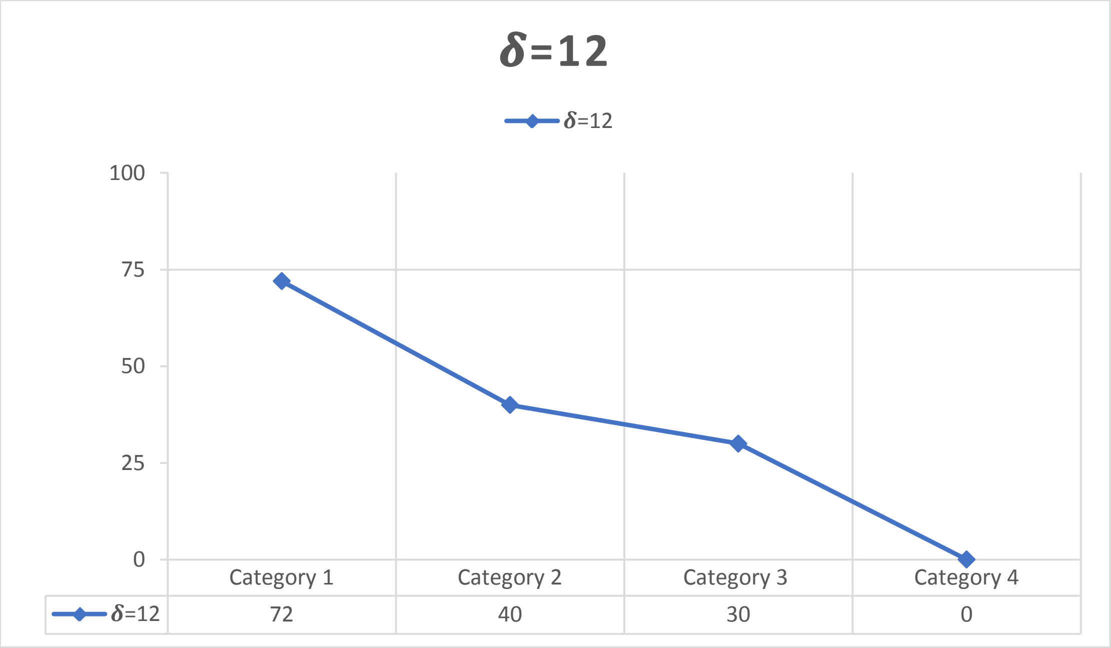


*Figure 4.16 : configuration of* $\boldsymbol{\delta}$*=12*

After selecting the best $\boldsymbol{\delta}$ which was *3* connected words, now we will start selecting the best $\boldsymbol{\varepsilon}$**:**

d) **Configuration ε**

We select the maximum token plagiarism threshold for any token to mark as plagiarized to be $\geq 75\%$ and this percent can be changed per Institution and can change on runtime. Therefore, after selecting the best **ε** which was $\geq 75\%$ from the total of the token, now we will start selecting the best $\boldsymbol{\zeta}$**:**

e) **Configuration** $\boldsymbol{\zeta}$

We select the maximum token plagiarism threshold for the whole file to mark as plagiarized to be $\geq 25\%$ and this percent can be changed per Institution and can change on runtime.

Now we have the Best result for each configurations as we see in Figure 4.17, so we will start our methodology of this numbers and let the user change them if they want.

**Configuration** $\boldsymbol{\beta = 12, \gamma = 3..n+1, \delta = 3..\beta, \varepsilon \geq 75\%, \zeta \geq 25\%}$

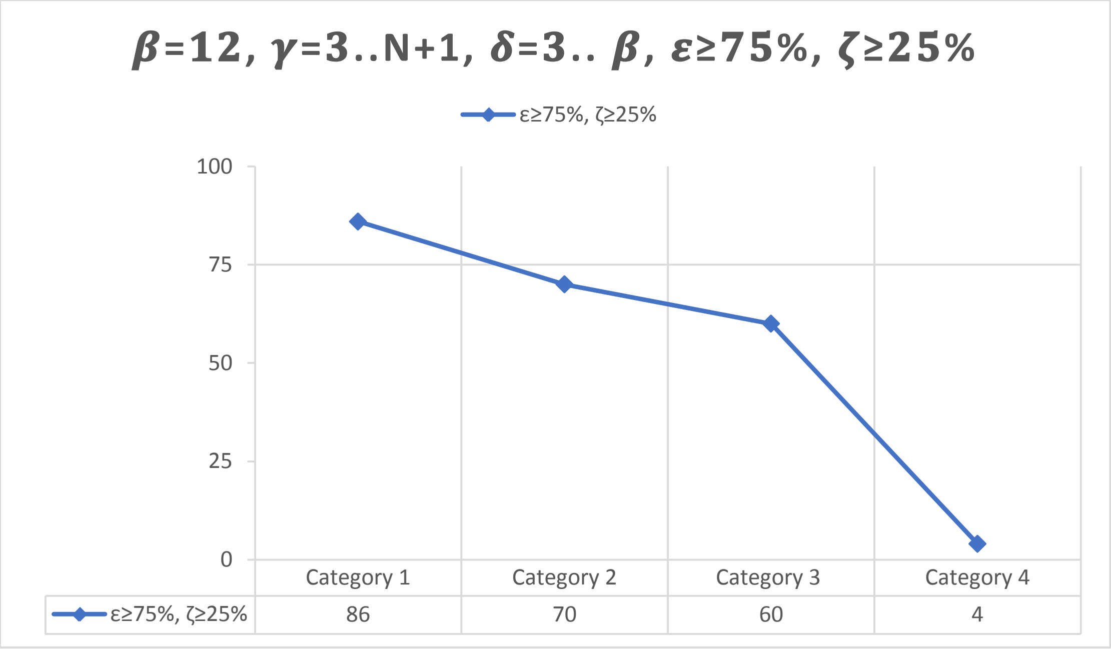
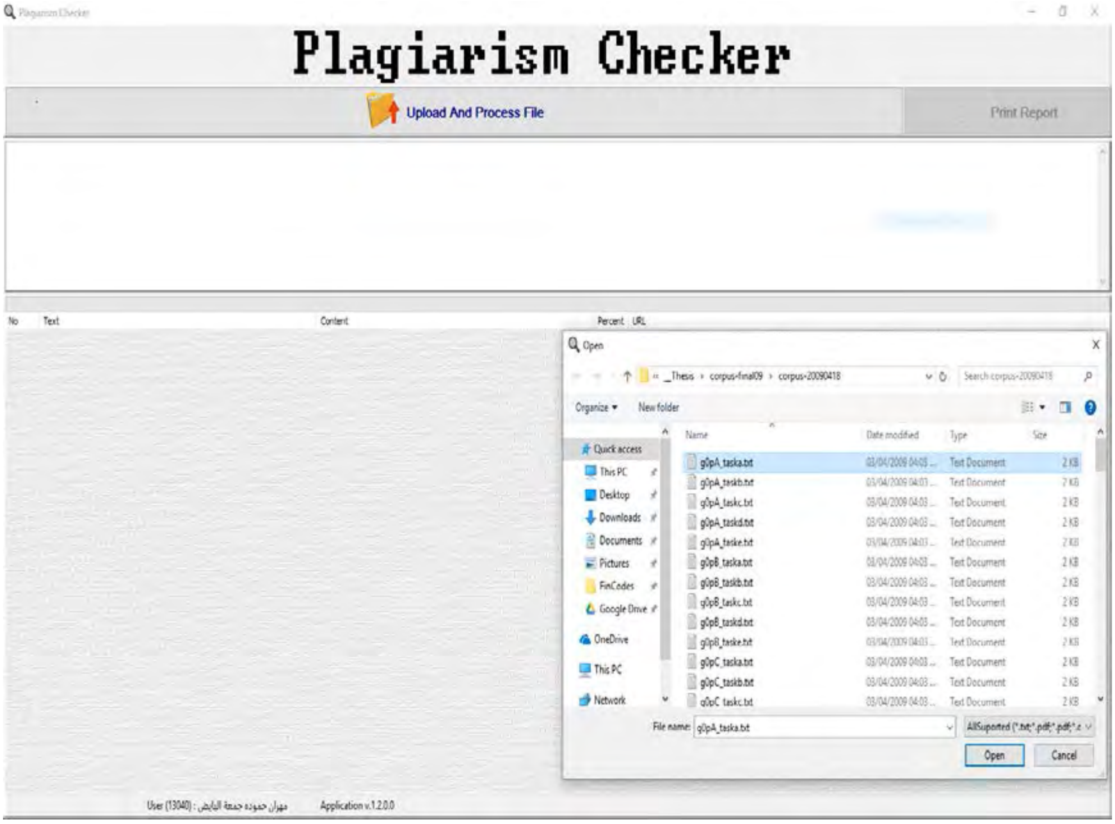


*Figure 4.17 : our final configuration for the system*

## 4.4. System Screenshot

Here we present our system screenshot after completing the implementation of the methodology.

Figure 4.18 illustrated the selecting the file for starting the process of detection.



| Name | Date modified | Type | Size |
|---|---|---|---|
| g0pA_taska.txt | 03/04/2009 04:05 ... | Text Document | 2 KB |
| g0pA_taskb.txt | 03/04/2009 04:03 ... | Text Document | 2 KB |
| g0pA_taskc.txt | 03/04/2009 04:03 ... | Text Document | 2 KB |
| g0pA_taskd.txt | 03/04/2009 04:03 ... | Text Document | 2 KB |
| g0pA_taske.txt | 03/04/2009 04:03 ... | Text Document | 2 KB |
| g0pB_taska.txt | 03/04/2009 04:03 ... | Text Document | 2 KB |
| g0pB_taskb.txt | 03/04/2009 04:03 ... | Text Document | 2 KB |
| g0pB_taskc.txt | 03/04/2009 04:03 ... | Text Document | 2 KB |
| g0pB_taskd.txt | 03/04/2009 04:03 ... | Text Document | 2 KB |
| g0pB_taske.txt | 03/04/2009 04:03 ... | Text Document | 2 KB |
| g0pC_taska.txt | 03/04/2009 04:03 ... | Text Document | 2 KB |
| g0pC_taskb.txt | 03/04/2009 04:03 ... | Text Document | 2 KB |
| g0pC_taskc.txt | 03/04/2009 04:03 ... | Text Document | 2 KB |

File name: g0pA_taska.txt

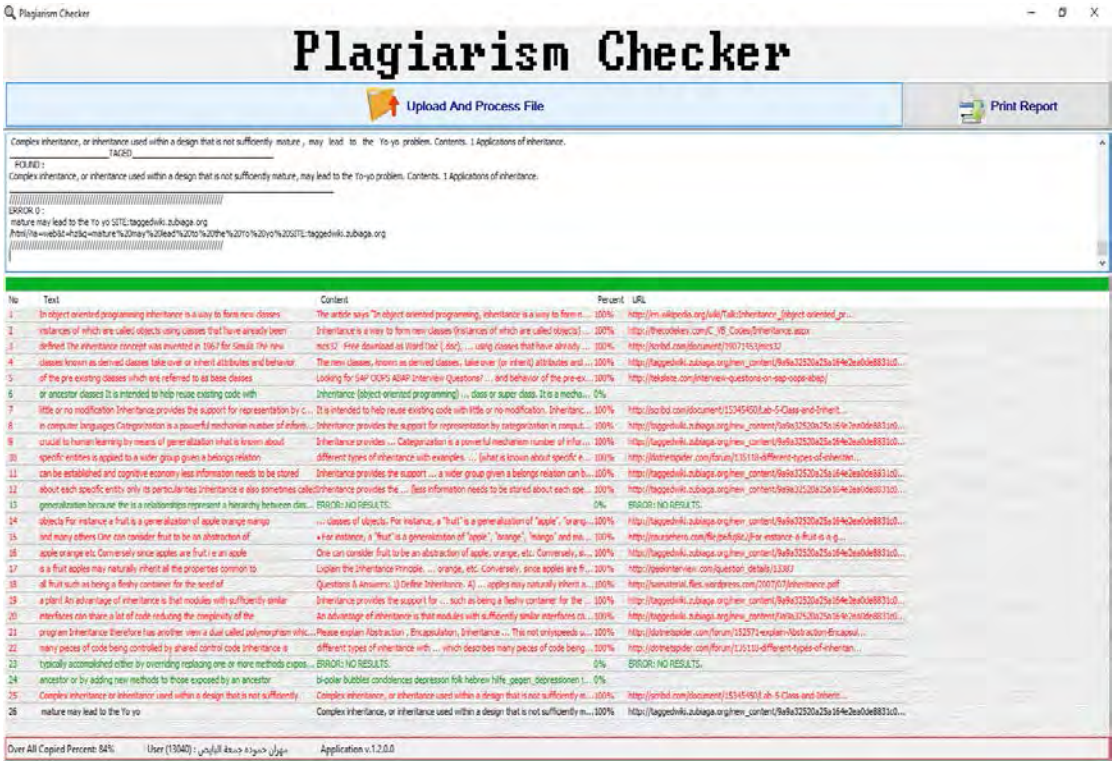
AllSuported (*.txt;*.pdf;*.pdf;*.z

User (13040) : مهران حمودة جمعة البايض

Application v.1.2.0.0

*Figure 4.18 : our system, selecting file for checking*

Figure 4.19 illustrated the system when they check token by token for polarized of not.



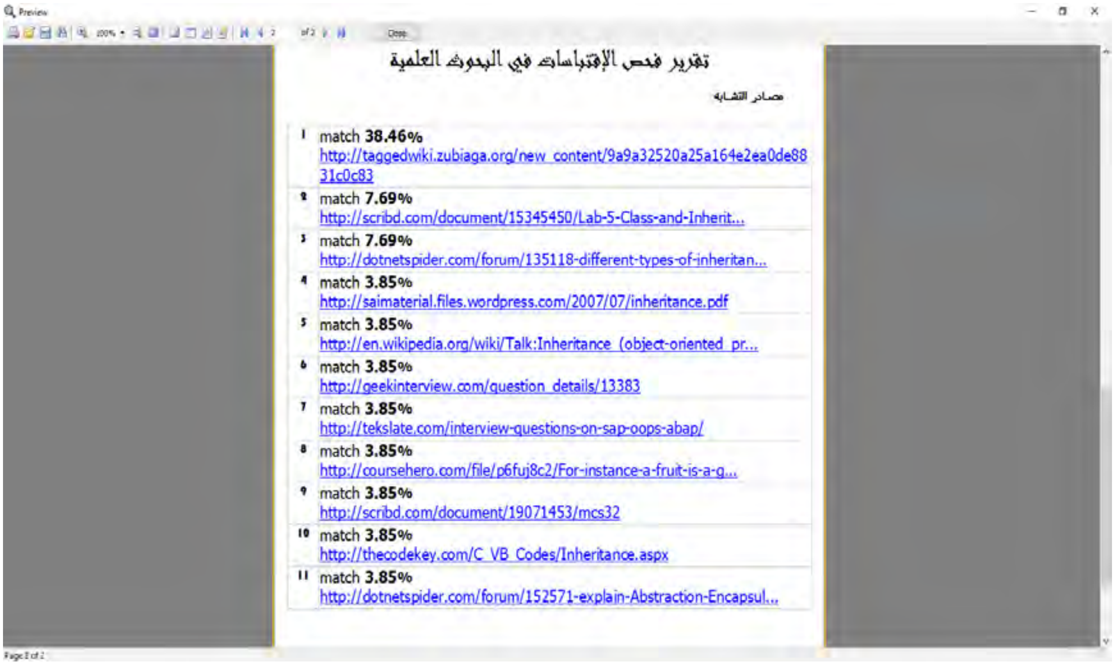
```
Complex inheritance, or inheritance used within a design that is not sufficently mature , may lead to the Yo-yo problem. Contents. 1 Applications of inheritance.
______________________TAGED______________________________
 FOUND :
Complex inheritance, or inheritance used within a design that is not sufficently mature, may lead to the Yo-yo problem. Contents. 1 Applications of inheritance.
__________________________________________________________________________
////////////////////////////////////////////////////////////////////
ERROR 0 :
 mature may lead to the Yo yo SITE:taggedwiki.zubiaga.org
/html?ie=web&t=hz&q=mature%20may%20lead%20to%20the%20Yo%20yo%20SITE:taggedwiki.zubiaga.org
////////////////////////////////////////////////////////////////////
```

| No | Text | Content | Percent | URL |
|---|---|---|---|---|
| 1 | In object oriented programming inheritance is a way to form new classes | The article says "In object oriented programming, inheritance is a way to form n... | 100% | http://en.wikipedia.org/wiki/Talk:Inheritance_(object-oriented_pr... |
| 2 | instances of which are called objects using classes that have already been | Inheritance is a way to form new classes (instances of which are called objects) ... | 100% | http://thecodekey.com/C_VB_Codes/Inheritance.aspx |
| 3 | defined The inheritance concept was invented in 1967 for Simula The new | mcs32 - Free download as Word Doc (.doc), ... using classes that have already ... | 100% | http://scribd.com/document/19071453/mcs32 |
| 4 | classes known as derived classes take over or inherit attributes and behavior | The new classes, known as derived classes, take over (or inherit) attributes and ... | 100% | http://taggedwiki.zubiaga.org/new_content/9a9a32520a25a164e2ea0de8831c0... |
| 5 | of the pre existing classes which are referred to as base classes | Looking for SAP OOPS ABAP Interview Questions? ... and behavior of the pre-ex... | 100% | http://tekslate.com/interview-questions-on-sap-oops-abap/ |
| 6 | or ancestor classes It is intended to help reuse existing code with | Inheritance (object-oriented programming) ... class or super class. It is a mecha... | 0% | |
| 7 | little or no modification Inheritance provides the support for representation by c... | It is intended to help reuse existing code with little or no modification. Inheritanc... | 100% | http://scribd.com/document/15345450/Lab-5-Class-and-Inherit... |
| 8 | in computer languages Categorization is a powerful mechanism number of inform... | Inheritance provides the support for representation by categorization in comput... | 100% | http://taggedwiki.zubiaga.org/new_content/9a9a32520a25a164e2ea0de8831c0... |
| 9 | crucial to human learning by means of generalization what is known about | Inheritance provides ... Categorization is a powerful mechanism number of infor... | 100% | http://taggedwiki.zubiaga.org/new_content/9a9a32520a25a164e2ea0de8831c0... |
| 10 | specific entities is applied to a wider group given a belongs relation | different types of inheritance with examples. ... (what is known about specific e... | 100% | http://dotnetspider.com/forum/135118-different-types-of-inheritan... |
| 11 | can be established and cognitive economy less information needs to be stored | Inheritance provides the support ... a wider group given a belongs relation can b... | 100% | http://taggedwiki.zubiaga.org/new_content/9a9a32520a25a164e2ea0de8831c0... |
| 12 | about each specific entity only its particularities Inheritance is also sometimes calle | Inheritance provides the ... (less information needs to be stored about each spe... | 100% | http://taggedwiki.zubiaga.org/new_content/9a9a32520a25a164e2ea0de8831c0... |
| 13 | generalization because the is a relationships represent a hierarchy between clas... | ERROR: NO RESULTS. | 0% | ERROR: NO RESULTS. |
| 14 | objects For instance a fruit is a generalization of apple orange mango | ... classes of objects. For instance, a "fruit" is a generalization of "apple", "orang... | 100% | http://taggedwiki.zubiaga.org/new_content/9a9a32520a25a164e2ea0de8831c0... |
| 15 | and many others One can consider fruit to be an abstraction of | • For instance, a "fruit" is a generalization of "apple", "orange", "mango" and ma... | 100% | http://coursehero.com/file/p6fuj8c2/For-instance-a-fruit-is-a-g... |
| 16 | apple orange etc Conversely since apples are fruit i e an apple | One can consider fruit to be an abstraction of apple, orange, etc. Conversely, si... | 100% | http://taggedwiki.zubiaga.org/new_content/9a9a32520a25a164e2ea0de8831c0... |
| 17 | is a fruit apples may naturally inherit all the properties common to | Explain the Inheritance Principle. ... orange, etc. Conversely, since apples are fr... | 100% | http://geekinterview.com/question_details/13383 |
| 18 | all fruit such as being a fleshy container for the seed of | Questions & Answers: 1) Define Inheritance. A) ... apples may naturally inherit a... | 100% | http://saimaterial.files.wordpress.com/2007/07/inheritance.pdf |
| 19 | a plant An advantage of inheritance is that modules with sufficiently similar | Inheritance provides the support for ... such as being a fleshy container for the ... | 100% | http://taggedwiki.zubiaga.org/new_content/9a9a32520a25a164e2ea0de8831c0... |
| 20 | interfaces can share a lot of code reducing the complexity of the | An advantage of inheritance is that modules with sufficiently similar interfaces ca... | 100% | http://taggedwiki.zubiaga.org/new_content/9a9a32520a25a164e2ea0de8831c0... |
| 21 | program Inheritance therefore has another view a dual called polymorphism whic... | Please explain Abstraction , Encapsulation, Inheritance ... This not onlyspeeds u... | 100% | http://dotnetspider.com/forum/152571-explain-Abstraction-Encapsul... |
| 22 | many pieces of code being controlled by shared control code Inheritance is | different types of inheritance with ... which describes many pieces of code being... | 100% | http://dotnetspider.com/forum/135118-different-types-of-inheritan... |
| 23 | typically accomplished either by overriding replacing one or more methods expos... | ERROR: NO RESULTS. | 0% | ERROR: NO RESULTS. |
| 24 | ancestor or by adding new methods to those exposed by an ancestor | bi-polar bubbles condolences depression folk hebrew hilfe_gegen_depressionen t... | 0% | |
| 25 | Complex inheritance or inheritance used within a design that is not sufficiently | Complex inheritance, or inheritance used within a design that is not sufficiently m... | 100% | http://scribd.com/document/15345450/Lab-5-Class-and-Inherit... |
| 26 | mature may lead to the Yo yo | Complex inheritance, or inheritance used within a design that is not sufficiently m... | 100% | http://taggedwiki.zubiaga.org/new_content/9a9a32520a25a164e2ea0de8831c0... |

Over All Copied Percent: 84%  User (13040) : مهران حمودة جمعة البايض  Application v.1.2.0.0

*Figure 4.19 : our system, checking document*

Figure 4.20 illustrated the final system report, which are having the similarities with the percent and a link to go to the matched document.



تقرير فحص الإقتباسات في البحوث العلمية

مصادر التشابه

| | |
|---|---|
| ١ | match 38.46%<br>http://taggedwiki.zubiaga.org/new_content/9a9a32520a25a164e2ea0de8831c0c83 |
| ٢ | match 7.69%<br>http://scribd.com/document/15345450/Lab-5-Class-and-Inherit... |
| ٣ | match 7.69%<br>http://dotnetspider.com/forum/135118-different-types-of-inheritan... |
| ٤ | match 3.85%<br>http://saimaterial.files.wordpress.com/2007/07/inheritance.pdf |
| ٥ | match 3.85%<br>http://en.wikipedia.org/wiki/Talk:Inheritance_(object-oriented_pr... |
| ٦ | match 3.85%<br>http://geekinterview.com/question_details/13383 |
| ٧ | match 3.85%<br>http://tekslate.com/interview-questions-on-sap-oops-abap/ |
| ٨ | match 3.85%<br>http://coursehero.com/file/p6fuj8c2/For-instance-a-fruit-is-a-g... |
| ٩ | match 3.85%<br>http://scribd.com/document/19071453/mcs32 |
| ١٠ | match 3.85%<br>http://thecodekey.com/C_VB_Codes/Inheritance.aspx |
| ١١ | match 3.85%<br>http://dotnetspider.com/forum/152571-explain-Abstraction-Encapsul... |

*Figure 4.20 : our system, report after checking for a doc*

## 4.5. Conclusion

In this chapter, we summarizes the experiments and the dataset that we tested on our methodology and we discuss the result of the system. In addition, we discuss the parameters that can be changed in our system and their best case for better detection percentage.

In the next chapter, we will summarizes the conclusion of the work we are completed on this research and what we will do in the future work.

# Chapter 5

# CONCLUSION

## 5.1. Conclusion

This Chapter summarizes the conclusion of the work completed on this research. We are conclude that the most plagiarism systems nowadays are responsible for storing the files and make a local repository internally to detect for the future documents. We have developed our method that use the enhanced features of searching that search engine provide without the need to store any file or the need to construct the local repository. On the same hand, we added the ability for searching inside the academic website that will use the system in order to optimize the output that they present for the public use. In addition, we presented support for Arabic document that have a few plagiarism detectors to use in the public, and made our system capable to use the test that contains Diacritics that make the plagiarism so hard in case of the Arabic synonyms.

## 5.2.Future Work

We achieved our result with the help of using DuckDuckGo in the process of detecting plagiarism on the documents, but we need to talk about some observation that we have seen during our work. The first observation if we use the search feature too fast the site will block our IP and no result will return. In addition, this will cost us to wait for every token and this mean more time in the searching; however, this is a key feature to fasten the searching of the document. In the other hand, we need to add other search engine sites and make our system capable of using more than one site and this will enhance the search results.

Another observation is the list of suspected documents that the system search and generate them over tokens, which need to be more flexible and make the users of the system optionally add site or files that they suspected.

In the other hand, what about the self-Plagiarized? If the author is the same as the author of the suspected document? How we can solve this; this problem will been solved by demonstrating a list like our suspected document and make our system exclude the result of any document that match our system.

The final observation we need to talk about is that our system can search for online document that the search engine can found. So we need to make sure that each file are uploaded and are searchable by the search engine and this will make the result perfect as we can, which can be solved by uploading each file to a site that can be searchable.

# REFERENCES

1. *Oxford Reference*. [cited 2017 9, 18]; Available from: http://www.oxfordreference.com/view/10.1093/oi/authority.20110803100329803.
2. Stamatatos, E., *Intrinsic plagiarism detection using character n-gram profiles.* threshold, 2009. **2**(1,500).
3. Zechner, M., et al. *External and intrinsic plagiarism detection using vector space models*. in *Proc. SEPLN*. 2009.
4. Seaward, L. and S. Matwin. *Intrinsic plagiarism detection using complexity analysis*. in *Proc. SEPLN*. 2009.
5. Alzahrani, S. and N. Salim, *Fuzzy semantic-based string similarity for extrinsic plagiarism detection.* Braschler and Harman, 2010: p. 1-8.
6. Ferrero, J., et al., *Deep Investigation of Cross-Language Plagiarism Detection Methods.* arXiv preprint arXiv:1705.08828, 2017.
7. Mcnamee, P. and J. Mayfield, *Character n-gram tokenization for European language text retrieval.* Information retrieval, 2004. **7**(1): p. 73-97.
8. Pataki, M., *A new approach for searching translated plagiarism.* 2012.
9. Barrón-Cedeno, A., et al. *On Cross-lingual Plagiarism Analysis using a Statistical Model*. in *PAN*. 2008.
10. Hilgert, L.W., et al. *Building Domain Specific Bilingual Dictionaries*. in *LREC*. 2014.
11. Gabrilovich, E. and S. Markovitch. *Computing semantic relatedness using wikipedia-based explicit semantic analysis*. in *IJcAI*. 2007.
12. Jinha, A.E., *Article 50 million: an estimate of the number of scholarly articles in existence.* Learned Publishing, 2010. **23**(3): p. 258-263.
13. *Google*. [cited 2017 9, 18]; Available from: https://www.google.com.
14. *DuckDuckGo*. [cited 2017 9, 18]; Available from: https://duckduckgo.com/.
15. *Top 15 Misconceptions About Turnitin* [cited 2017 9, 18]; Available from: http://turnitin.com/en_us/resources/blog/421-general/1644-top-15-misconceptions-about-turnitin.
16. Robertson, S., *Understanding inverse document frequency: on theoretical arguments for IDF.* Journal of documentation, 2004. **60**(5): p. 503-520.
17. Kang, N., A. Gelbukh, and S. Han. *PPChecker: Plagiarism pattern checker in document copy detection*. in *International Conference on Text, Speech and Dialogue*. 2006. Springer.
18. Jiffriya, M., et al. *AntiPlag: Plagiarism detection on electronic submissions of text based assignments*. in *Industrial and Information Systems (ICIIS), 2013 8th IEEE International Conference on*. 2013. IEEE.
19. Bensalem, I., P. Rosso, and S. Chikhi. *Intrinsic plagiarism detection in Arabic text: Preliminary experiments*. in *II Spanish Conference on Information Retrieval (CERI'12)*. 2012.
20. Menai, M.E.B., *Detection of plagiarism in Arabic documents.* International journal of information technology and computer science (IJITCS), 2012. **4**(10): p. 80.
21. *Turnitin*. [cited 2017 9, 18]; Available from: http://turnitin.com/.
22. Dahl, S., *Turnitin® The student perspective on using plagiarism detection software.* Active Learning in Higher Education, 2007. **8**(2): p. 173-191.
23. *Delphi*. [cited 2017 9, 18]; Available from: https://www.embarcadero.com/products/delphi.
24. *Office*. [cited 2017 9, 18]; Available from: https://www.office.com/.

25. *OLE* [cited 2017 9, 18]; Available from: http://searchwindowsserver.techtarget.com/definition/OLE-Object-Linking-and-Embedding.
26. *Wikipedia*. [cited 2017 9, 18]; Available from: https://www.wikipedia.org/.
27. *Wikia*. [cited 2017 9, 18]; Available from: http://www.wikia.com/fandom.
28. *CrunchBase*. [cited 2017 9, 18]; Available from: https://www.crunchbase.com/.
29. *GitHub* [cited 2017 9, 18]; Available from: https://github.com/.
30. *WikiHow* [cited 2017 9, 18]; Available from: https://www.wikihow.com/Main-Page.
31. *The Free Dictionary*. [cited 2017 9, 18]; Available from: https://www.thefreedictionary.com/.
32. *DuckDuckGo API*. [cited 2017 9, 18]; Available from: https://duckduckgo.com/api.
33. *DuckDuckGo Syntax*. [cited 2017 9, 18]; Available from: https://duck.co/help/results/syntax.
34. *HTML*. [cited 2017 9, 18]; Available from: https://www.w3schools.com/html/html_intro.asp.
35. *DIHtmlParser*. [cited 2017 9, 18]; Available from: https://www.yunqa.de/delphi/products/htmlparser/index.
36. *JSON*. [cited 2017 9, 18]; Available from: https://www.w3schools.com/js/js_json_intro.asp.
37. *SuperObject*. [cited 2017 9, 18]; Available from: https://github.com/hgourvest/superobject.
38. *Regular expression* [cited 2017 9, 18]; Available from: http://www.oxfordreference.com/view/10.1093/oi/authority.20110803100411512.
39. *PerlRegEx* [cited 2017 9, 18]; Available from: https://www.regular-expressions.info/delphi.html.
40. Clough, P. and M. Stevenson, *Developing a corpus of plagiarised short answers*. Language Resources and Evaluation, 2011. **45**(1): p. 5-24.
41. Potthast, M., et al. *An evaluation framework for plagiarism detection*. in *Proceedings of the 23rd international conference on computational linguistics: Posters*. 2010. Association for Computational Linguistics.
42. Abu Naser, S. S. (2008). Developing an intelligent tutoring system for students learning to program in C++. Information Technology Journal, 7(7), 1055-1060.
43. AbuEloun, N. N., & Abu Naser, S. S. (2017). Mathematics intelligent tutoring system. International Journal of Advanced Scientific Research, 2(1), 11-16.
44. Abu-Naser, S., Ahmed, A., Al-Masri, N., Deeb, A., Moshtaha, E., & AbuLamdy, M. (2011). An intelligent tutoring system for learning java objects. International Journal of Artificial Intelligence and Applications (IJAIA), 2(2).
45. Abu-Naser, S. S., El-Hissi, H., Abu-Rass, M., & El-Khozondar, N. (2010). An expert system for endocrine diagnosis and treatments using JESS. Journal of Artificial Intelligence; Scialert, 3(4), 239-251.
46. Alawar, M. W., & Abu Naser, S. S. (2017). CSS-Tutor: An intelligent tutoring system for CSS and HTML. International Journal of Academic Research and Development, 2(1), 94-98.
47. Al-Bastami, B. G., & Abu Naser, S. S. (2017). Design and Development of an Intelligent Tutoring System for C# Language. EUROPEAN ACADEMIC RESEARCH, 6(10), 8795.
48. Al-Bayed, M. H., & Abu Naser, S. S. (2017). An intelligent tutoring system for health problems related to addiction of video game playing. International Journal of Advanced Scientific Research, 2(1), 4-10.

49. Aldahdooh, R., & Abu Naser, S. S. (2017). Development and Evaluation of the Oracle Intelligent Tutoring System (OITS). EUROPEAN ACADEMIC RESEARCH, 6(10), 8711-8721.

50. Alhabbash, M. I., Mahdi, A. O., & Abu Naser, S. S. (2016). An Intelligent Tutoring System for Teaching Grammar English Tenses. EUROPEAN ACADEMIC RESEARCH, 6(9), 7743-7757.