



PARAPHRASE BASED PLAGIARISM DETECTION SYSTEM USING GENSIM AND NLTK PYTHON LIBRARY

*A Thesis Submitted in Partial Fulfillment of the Requirements for the
award of the degree of*

Bachelor of Science in Electrical and Computer Engineering
(Computer Engineering)

By

Name of students

ID

BEREKET HAILUETS0187/08

BIRTUKAN NEGA.....ETS0245/08

Under the guidance of

Miss. Selam (MSc)

ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY

Jan 2022

Examining Committee Approval Sheet

Title of Thesis: paraphrase based plagiarism detection system using GENSIM and NLTK python library

Student Names (& ID):

1. Bereket Hailu	ETS0187/08
2. Birtukan Nega	ETS0245/08

Approved by the examining committee members:

	Name	Academic Rank	Signature	Date
Advisor:	_____	_____	_____	_____
Co-Advisor:	_____	_____	_____	_____
Examiner:	_____	_____	_____	_____
Examiner:	_____	_____	_____	_____

	Name	Signature	Date
DC Chairperson:	_____	_____	_____
Associate Dean for Under Graduate Programs:	_____	_____	_____

Acknowledgement

Firstly we would like to thank our God for his never-ending grace, mercy and for keeping us safe from the current disease covid-19 lockdown and helping us to pass all obstacles we encounter through the time of education and thesis work. Next we would like to thank our technical advisor Miss. Selam for her willingness to be an advisor and coach our work. Finally we appreciate our fellows and class mate for your support at different aspects.

Abstract

As the technology is growing very fast and usage of computer systems is increasing compared to old times, Plagiarism is the main issue which is increasing today. Wrongful appropriation of someone else's work is known as plagiarism. We try to contribute in detecting plagiarism by making our senior year project in this area by developing paraphrase based plagiarism detection system using generate similar (GENSIM) and NLTK python library. It is a web-based application which can detect plagiarized content after cleaning unstructured text document using NLTK preprocessing techniques like sentence segmentation, tokenization, stemming, lemmatization etc. We have used GENSIM library for statistical machine learning to perform word vectors, to compute similarity of text and classify it using TF-IDF module.

Table of contents

Contents

Acknowledgement	iii
Abstract	iv
Table of contents	v
List of Figures	vii
List of Tables	vii
Acronyms	viii
CHAPTER-1	1
INTRODUCTION	1
1.2 Statement of the Problem	3
1.3 Objectives	4
1.3.1 General Objective	4
1.3.2 Specific Objectives	4
1.4. Significance of the Project	5
1.5 Scope and Limitation of the Project	5
CHAPTER-2	6
LITERATURE REVIEW	6
2.1 Introduction	6
2.2 History of Plagiarism Detection Approaches	7
2.3 Extrinsic vs. Intrinsic Methods for Plagiarism Detection	7
2.4 Extrinsic Methods for Plagiarism Detection	7
2.4.1 String/text matching technique	8
2.4.2 Semantic based detection technique	8
2.5 Intrinsic Methods for Plagiarism Detection	8
2.6 Plagiarism Detection Tools	9
2.7 Summary	12
CHAPTER-3	13
METHODOLOGY AND SYSTEM DESIGN	13
3.1 Methods	13
3.2 System Design with Block Diagrams	14
3.4 Description of Components (Technologies) and Specifications	16

3.4.1 Natural Language Processing (NLP)	16
3.4.2 Natural Language Toolkit	17
3.4.3 Gensim	17
3.4.4 Google Search	17
3.4.5 Web Scraping	18
3.4.6 Bing Web Search API	18
3.4.7 TF-IDF	18
3.4.8 How to compute TF*IDF	19
CHAPTER-4	21
RESULTS AND DISCUSSIONS	21
4.1 Result	21
CHAPTER-5	23
CONCLUSIONS AND SCOPE FOR FUTURE WORK	23
5.1 Conclusions	23
5.2 Scope for Future Work	23
REFERENCE	24
APPENDIX – I	25

List of Figures

Figure 1 Plagiarism Types, detection system and techniques	2
Figure 2 General methods.....	4
Figure 3 General Architecture of Extrinsic PDS.....	7
Figure 4 Text mining pipeline.....	13
Figure 5 proposed system block diagram	14
Figure 6 Natural Language Processing (NLP).....	17
Figure 7 Home Page	21
Figure 8 Input Receivers.....	22
Figure 9 Output	22

List of Tables

Table 1 Top 10 free plagiarism detection tools	7
--	---

Acronyms

AI: Artificial Intelligence

API: Application Programming Interface

BOW: Bag of words

IR: Information Retrieval

GENSIM: Generate Similar

NLP: Natural Language processing

NLTK: Natural Language Toolkit

PDS: Plagiarism Detection System

RESTful: Representational state transfer

TF-IDF: Term Frequency Information Document Frequency

SWR: Stop Words Removal

URL: Uniform Resource Locator

CHAPTER-1

INTRODUCTION

Nowadays, due to the digital world or quick development of the web resources, the internet has facilitated immediate information access and this also led to large amounts of unstructured data, especially text. A major drawback of the easy information access made possible by the Internet is the widespread prevalence of the phenomenon of copying and reusing information without permission.

Plagiarism can be termed as the unauthorized reuse of content or ideas without giving the credit to the original authors. Plagiarism is a major threat to academics and has to be suitably addressed to ensure integrity and authenticity. Literal plagiarism includes copy–paste operations and is usually easy to detect. More sophisticated forms of plagiarism may involve translation, summarization, and paraphrasing and are more difficult to recognize. One of the most difficult to detect and relatively less addressed forms is paraphrase plagiarism in which the original content may be completely reworded and altered considerably. Plagiarism detection systems focus on ensuring the originality of text content

The objective of this work is to investigate the suitability of utilizing a machine learning-based paraphrase recognition system for plagiarism detection on TF–IDF technique of GENSIM library and NLTK techniques

1.1 Background

There are different types of plagiarism with respect to gravity and frequency. The general classification of plagiarism types, detection systems and techniques likes in the figure below:

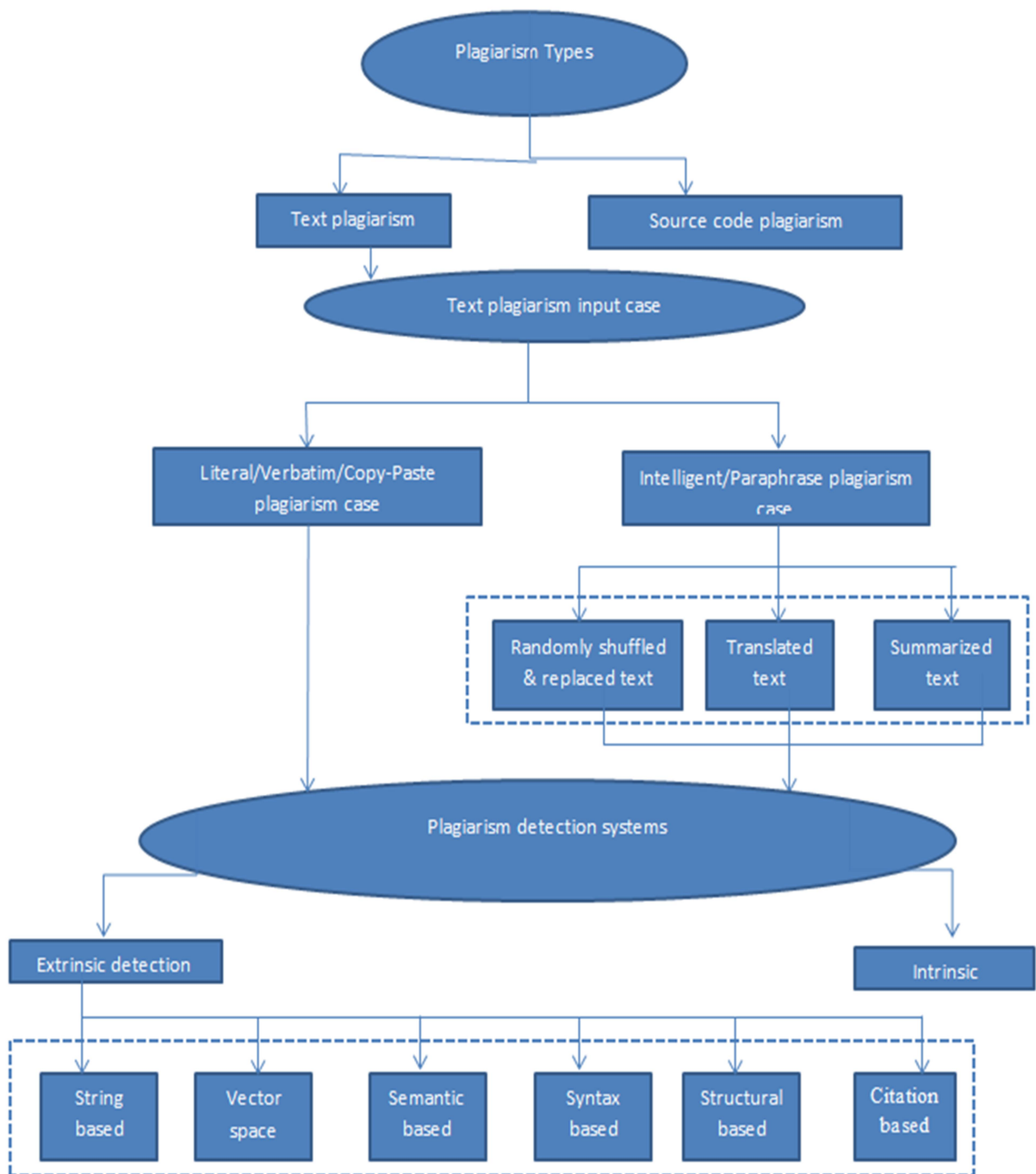


Figure 1 Plagiarism Types, detection system and techniques

1.2 Statement of the Problem

The ease of sharing online information in this age of digital communication has encouraged the misuse of text and the prevalence of plagiarism. Academic bodies and scientific publishing companies are playing an active role in detecting plagiarism in order to maintain the integrity of academic publications.

In [14] Redfern and Barnwell (2009) pointed out that many cases of academic work submitted by students contain some level of plagiarized material. [15] Roig (2001) reported that up to 60% of student assignments contain some level of plagiarized and paraphrased material. Few years later, the international center of academic integrity (ICAI) published that 86% of students were involved in some form of plagiarism.

A wide range of studies focused on researching and developing methods for effectively combating plagiarism. However, there is lack of effective plagiarism detection methods.

In this plagiarism detection system project we are going to build a web-based application which can detect plagiarized content by checking the similarities of text using Gensim and NLTK python library, which is a branch of artificial intelligence (AI) and the most popular library for NLP.

It contains text preprocessing techniques like tokenization, stop of word removal, breaking text into phrases and search for similarity after that stem & lemmatize finally convert into vector form using term frequency inverse document frequency (TF-IDF) and retrieve the similarity percentage for each URL using general method as shown below.

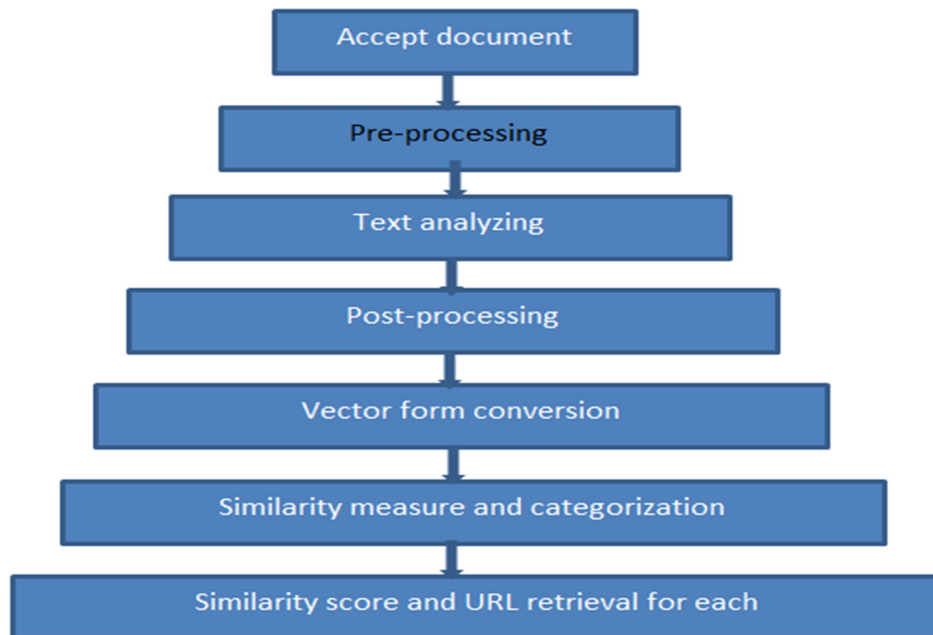


Figure 2 General methods

1.3 Objectives

1.3.1 General Objective

Developing web based plagiarism detection system using natural language toolkit (NLTK) python library and Gensim.

1.3.2 Specific Objectives

- To develop an intelligence approach this can detect plagiarism using machine learning.
- To investigate the role of machine learning approaches for plagiarism detection based on the statistical measure of the text.
- To apply the use of NLP on plagiarism detection system
- To evaluate the proposed plagiarism detection techniques

1.4. Significance of the Project

Plagiarism detection system is one of the most important issues to academic enterprises like universities, schools & institutions, journals, research center and conferences; plagiarism detection and prevention became one of the educational challenges.

There are big numbers of plagiarism software used for plagiarism detection and many of detection tools have been developed by researchers but still they have some limitations as they cannot understand the context of text they only check the similarity of exact word or phrase in this project we try to solve this problem by using natural language processing technique in order to understand the text from the context and check for similarity using search engine API and retrieve for similarity percentages and the site from which URL plagiarized.

1.5 Scope and Limitation of the Project

Scope: The main goal of this project is to change current detection system which works on by comparing only similar words or sentences into intelligent system using machine learning

- ✚ This project will cover plagiarism detection in other languages except Amharic language.
- ✚ It uses TF*IDF algorithm in order to convert text into vector form and GENSIM to compute similarity.
- ✚ It uses search engine API (Bing search API).
- ✚ It retrieves the similarity percentage for each similar URL and renders.
- ✚ It can accept up to 1000 words once and can check many times per day.

Limitations:

- It do not have language translation (i.e. monolingual) checks for single language only
- The speed of the system depends on the speed of internet, the system we use to run and it was affected by the sleep method we have used to avoid blocking come from Google.

CHAPTER-2

LITERATURE REVIEW

2.1 Introduction

In this chapter we will see a number of existing approaches which aimed to address the problem of plagiarism with varying degrees of success and also we will discuss the strengths and limitations of the existing approaches. There are various works on plagiarism detection so far most of the systems nowadays use the text manipulation system techniques and others use machine learning based and AI based methods of detection. Some of the methods consist of character based, structural based, cross language based, syntax based, cluster-based method, semantic based method and citation-based methods. Various tools are using those methods. Some graduate students in India bring new approaches on how to detect plagiarism using K-NN (K-nearest neighbor machine learning techniques and this technique. Using k-means algorithm they compare the categorized data set with the document in the training set K-NN can be memory-based learning or instance-based learning. Sometimes using this technique causes some difficulties such as falling off words in the wrong category. And others use Euclidean distance, cosine measure etc. The other technique is text-based systems this technique is using parsing and how the text set is classified into tokens. First parsed string and compare with others. Then compare query files with stored files. Using some conditions, they set their percentage of each copied or similar word. Using this method, they find the frequency of each repeated or similar word and use some threshold to decide whether the document is plagiarized or not.

There are now more sophisticated software tools to identify plagiarism, which use “pattern-matching technology.” They include Turnitin, Veriguide, Plagiarism Checker, Grammarly plagiarism checker, PlagiarismCheckerX, Plagiarisma, Crosscheck, iThenticate, SafeAssign, anti-plagiarism, Plagium, Plagscan, and Write check. Built-in text matcher software is also in use. There’s also another software tool to check computer programs, called MOSS (Measure of Software Similarity).

The literature refers to a wide range of plagiarism conduct types, starting from a simple copy and paste of the exact piece of text to the imitation of ideas, summaries or obfuscation by using

translation [1]. In extrinsic plagiarism detection there are challenges like text manipulation practices such as using synonyms of words, transferring from active to passive or vice versa. However, in case of intrinsic plagiarism detection the only challenges that faces are imitation and obfuscation

2.2 History of Plagiarism Detection Approaches

The recent developments in related fields such as machine learning, data mining, computational linguistics, and information retrieval (IR) has impacted the research on automated plagiarism detection in written texts

2.3 Extrinsic vs. Intrinsic Methods for Plagiarism Detection

From chapter 1 point of view as shown in figure 1 in case of text plagiarism detection, there are mainly two formal tasks which are extrinsic/external detection and intrinsic/internal detection. In extrinsic plagiarism detection the suspected documents are compared against a reference source corpus. whereas in intrinsic PDS, a reference corpus is unavailable and the suspicious document is analyzed single-handedly without being compared with any sources. In extrinsic PDS, various detection techniques can be employed such as string based, syntax based, semantic based, structural based and citation based techniques or a combination of these techniques.

2.4 Extrinsic Methods for Plagiarism Detection

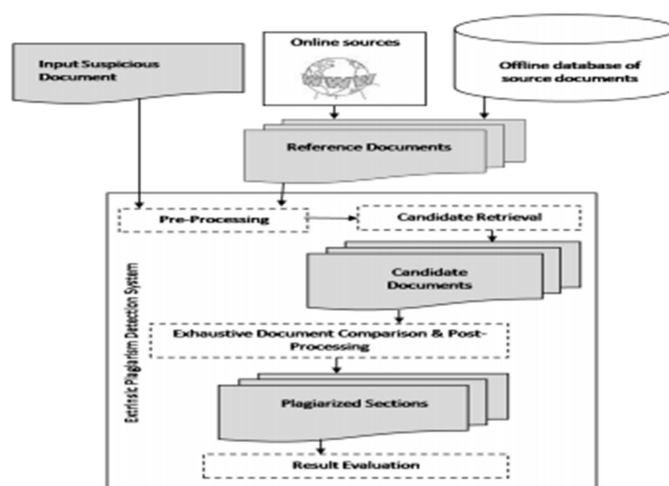


Figure 3 General Architecture of Extrinsic PDS

2.4.1 String/text matching technique

It uses the texts/strings of documents to check for similarity. The system was not able to detect plagiarism based on multiple sources. [10] Extracted the contextual and surrounding N-grams which are extended N-gram models. They used sorted word 3-grams and sorted word 1-skip-3-grams. The accuracy dropped as the paraphrasing complexity increased.

All these detection systems were effective for detecting plagiarism cases with simple copy-paste and intelligent plagiarism cases with small random shuffling while the efficiency of detection dropped as plagiarism complexity increased. In general, N-gram based models were found to be less effective when it comes to complex obfuscation types. But the exhibition of good precision shows its potential to be combined and used in hybrid approaches

2.4.2 Semantic based detection technique

[11] Have emphasized the urgent need for sufficient algorithms that are able to capture semantic patterns between two texts. A study by [12] determined that existing detection tools are miscarried in detecting obfuscated text due to their algorithms' limitations. The similarity techniques of these algorithms do not take the linguistic or semantic structure into consideration during comparison procedures. The research in this 40 thesis has benefited from the work that was conducted by [12] and followed the recommendation of integrating linguistic statistics and semantics to develop a new approach for extrinsic plagiarism detection. Another study by [10] proposed a technique based on sentences to compare between two sentences using their words and corresponding synonyms. This method attempted to detect semantics; however, it needs to be improved to capture the correct meaning as not all synonyms relate to every meaning.

2.5 Intrinsic Methods for Plagiarism Detection

The intrinsic methods for plagiarism detection differ from extrinsic methods as they do not use a references collection to compare with. This type of detection method relies on capturing the variations in written text by extracting the syntactic and lexical features. Then a comparison is performed between the suspicious text and the same author's work in order to identify the variation patterns.

2.6 Plagiarism Detection Tools

In the past two decade, several plagiarism detection tools have been developed, although we will discuss some of these tools the top 10 free plagiarism detection tools (13) that will allow eLearning professionals to tackle the plagiarism nightmare in brief in the following table.

Table 1 Top 10 free plagiarism detection tools

No.	Tools	Features	Pros	Cons
1	Dupli Checker	Most effective plagiarism detection tool on internet	<ul style="list-style-type: none">● Free of charge, Ease of access, 50 scans per day for registered users● 2 ways to check (either copy and paste or upload a Doc/text file)	1search per day for unregistered users
2	Copyleaks	Cloud-based authentication platform enables you to track how eLearning content is being used all around the Internet.	<ul style="list-style-type: none">● For education & business● Multiple file format and language● Variety of tools eg. Allows you to use the API tool to search for plagiarized content all over the Internet.	You can use it only after signing up. Free page restriction (10 page for free)
3	PaperRater	A multi-purpose free plagiarism detection tool that is used in over 140 countries	<ul style="list-style-type: none">● 3 Tools In 1: Proofreader & Grammar Check, Vocabulary Builder, Plagiarism Checker● Developed By Industry Experts	No ability to save reports

			<ul style="list-style-type: none"> ● Fast Results 	
4	Plagiarisma	Free online checker, multi-purpose detection tool i.e. used by students, teachers, writers, as well as various members	<ul style="list-style-type: none"> ● 190+ Languages Supported ● Plagiarism check by URL, online/file upload ● Supported documents - TXT, HTML, RTF, DOC, DOCX, PDF. ● Outputs the text as Unique if not plagiarized 	Limited plagiarism checks per day
5	Plagiarism Checker	User-friendly, entirely free plagiarism detection tool	<ul style="list-style-type: none"> ● Detailed Guidelines, entirely online ● Check If Others Have Plagiarized Your Online Content:- Click on the “For authors” option to check whether they have plagiarized your content and posted it on the Internet. You may also get a notification by email. 	Supports only Google /Yahoo browsers
6	Plagium	Basic but fully functional free detection tool with different levels of	<ul style="list-style-type: none"> ● Easy to use ● It features 2 types of searches, quick search and deep 	Limited Free Features

		search.	search. <ul style="list-style-type: none"> ● Free for up to 5,000 characters each time 	
7	PlagScan	For both individuals and businesses that checks texts against online content, scientific journals and the user's documents as well	<ul style="list-style-type: none"> ● 3 Ways To Do Plagiarism Checks: <ul style="list-style-type: none"> a) directly pasting you text into the appropriate field, b) importing the file from the web by entering its URL at the indicated area, or uploading it from a cloud storage area such as Drop box, Google Drive, or OneDrive, c) Uploading a file from your desktop. ● No Subscriptions For Private ● Integration Features 	Relatively Complicated Interface
8	PlagTracker	Fast free plagiarism detection tool that searches both websites and academic databases by copying and pasting text, or file uploading	<ul style="list-style-type: none"> ● Addressing To Different User Groups (Students, teachers, publishers and site owners) ● Detailed Reports-the user gets informed as to what parts need to be cited and a list of sources to be used ● 6 Languages 	No file uploading in free version

			Supported Check your content in English, French, Spanish, German, Romanian and Italian	
9	Quetext	Basic layout and functional interface that checks against the Internet, as well as various databases.	<ul style="list-style-type: none"> • No subscription -it's entirely free • Unlimited usage-no account, registration or download needed 	No file uploading
10	Plagiarismhunter	Online plagiarism checker that checks with 5 different plagiarism software systems with one click.	<ul style="list-style-type: none"> • You may scan text with 5 different plagiarism checkers on one website and see which one will show the best results. • You can try to scan for free with one free tool. • No need for registration, passwords 	Slower check speed. While other tools will check your paper in 2 minutes, this tool will need around 4 minutes. Ad hoc check. You can't collect your previously checked papers and get back to them later, as papers will be deleted after 24 hours

2.7 Summary

There are various works on plagiarism detection so far most of the systems nowadays the common problem noted with most of these tools are with their lack of ability to detect intelligent manipulations, even though they claim to be. In today's world, with the ease of access to online translation and summarization tools a plagiarist can easily perform intelligent and complex manipulations in source text which can surpass the detection capacity of these tools.

CHAPTER-3

METHODOLOGY AND SYSTEM DESIGN

3.1 Methods

Our plagiarism detection system uses text analyzer in order to extract machine-readable information from unstructured text to enable data driven approach towards managing content, to overcome the ambiguity of human language and achieve high accuracy for a specific domain. It requires development of customized text mining pipelines.

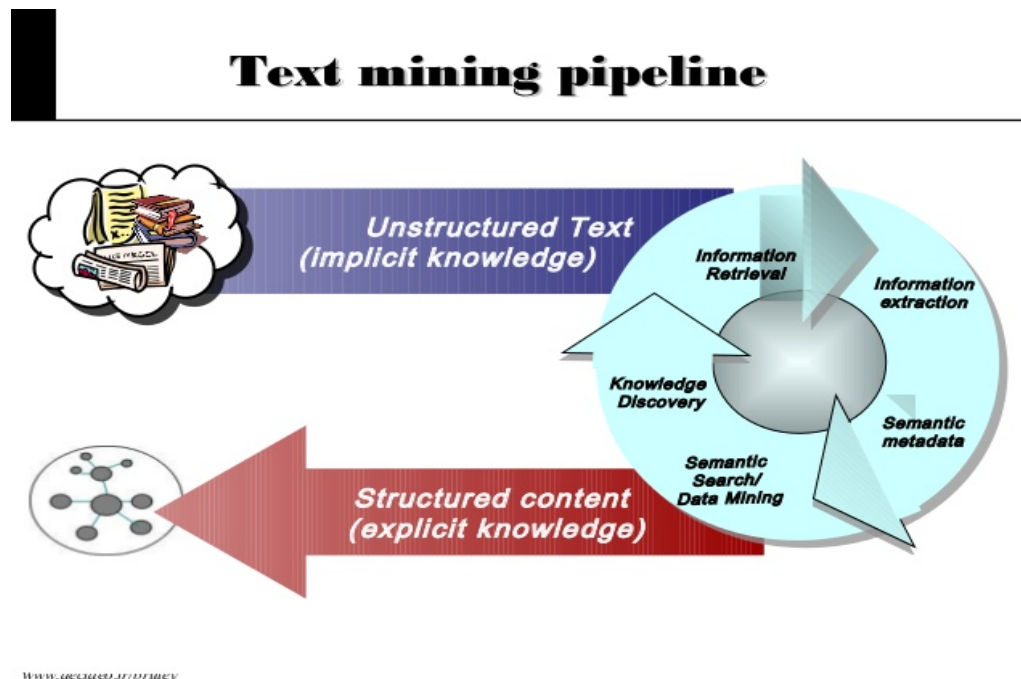


Figure 4 Text mining pipeline

It receives text document and then preprocess that the text by applying NLP techniques like tokenization, stop word removal, stemming, and lemmatization. After cleaning the unstructured data it converts to vector form to make it machine readable using TF-IDF and we have used GENSIM similarity to measure similarities of processed text documents. Finally it retrieves similarity scores for each similar URL and has a rendering feature that opens similar sites

3.2 System Design with Block Diagrams

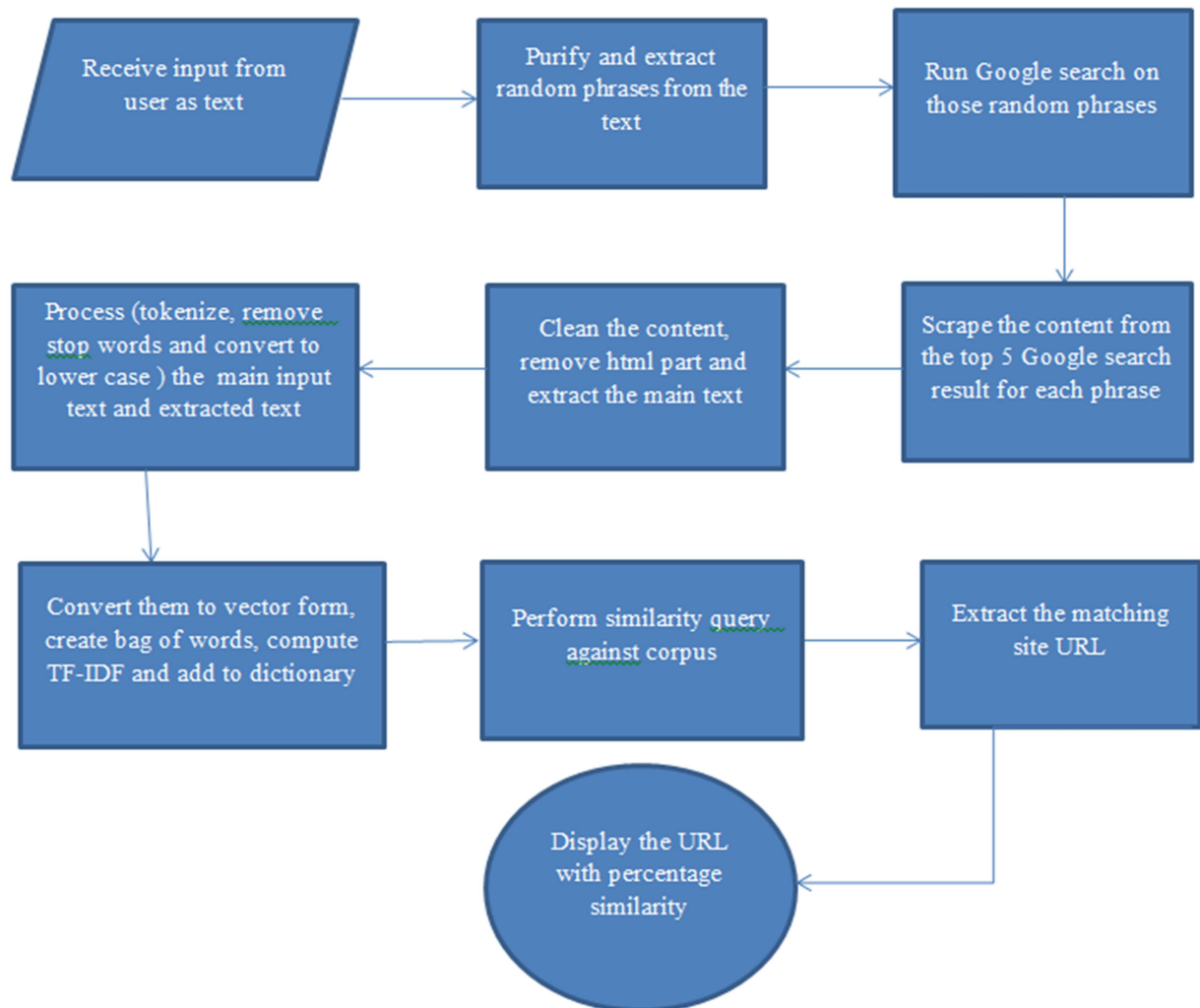


Figure 5 proposed system block diagram

Step1

In the first step the system accepts an input from users in the form of either by uploading in file formats of (.txt, .pdf, .docx) or copy paste as we want. Then if it is in the uploaded form, it extracts the content of file immediately and posts the extracted text into the text area provided in order to make suitable for computation purpose.

Step2

The first preprocessing segmentation starts here by purifying and extracting random phrases of an input document according to the posted input in the 1st step and removing the spaces, numbers, unwanted digit and symbols, which makes suitable and easy for the next process.

Step3

In this case it uses web scrapper Bing Web Search API to get similar documents of an input data by running Google search on those random phrases on www.bing.com . In this stage we encounter a problem that Goggle continually blocked an immediate searching for large data so to solve the problem we have used sleep function on the searching loop function.

Step4

After making search on those each segmented phrase it scrapes the URL of top 5 Google searches results for each phrase.

Step5

Then it extracts the main content of each URL by cleaning and removing the html part.

Step6

Post-process the extracted content and the user input simultaneously. It tokenizes, remove stop words and convert in to lower cases to make suitable for the next step of vector conversion and similarity computation.

Step7

After post-processing both input of user and the extracted contents will be converted to vector form, created a bag of words, TF-IDF is computed for each word and added to the dictionary in order to make suitable for similarity computation.

Step8

Perform similarity query against corpus and compute the similarity.

Step9

It puts the matching site URL with its computed similarity percentage in the dictionary after computing the similarity.

Step10

Finally the system will display URL for each similar site with similarity percentage and we can open the detected site content.










3.4 Description of Components (Technologies) and Specifications

Here are the components and technologies that we have been used for the development of our Plagiarism detection system:

3.4.1 Natural Language Processing (NLP)

First **natural language** refers to a human language such as English, Russian, German, or Japanese as distinct from the typically artificial command or programming language with which one usually talks to a computer and **Processing** how computers carries out instructions then **NLP** deals with simply how to deal with text.

Natural Language Processing (NLP) is a field of artificial intelligence that enables computers to analyze and understand (process) human language like speech or text in order to perform different tasks like:

-  Information Retrieval (Google finds relevant and similar results).
-  Information Extraction (Gmail structures events from emails).
-  Machine Translation (Google translate translates language from one language to another).
-  Virtual assistants (Google Assist, Siri, and Alexa).
-  Question Answering (IBM Watson's answers to a query).
-  Sentiment Analysis
-  Spam Filter (Gmail filters spam emails separately).
-  Auto-Predict (Google Search predicts user search results).
-  Auto-Correct (Google Keyboard and Grammarly correct words otherwise spelled wrong).

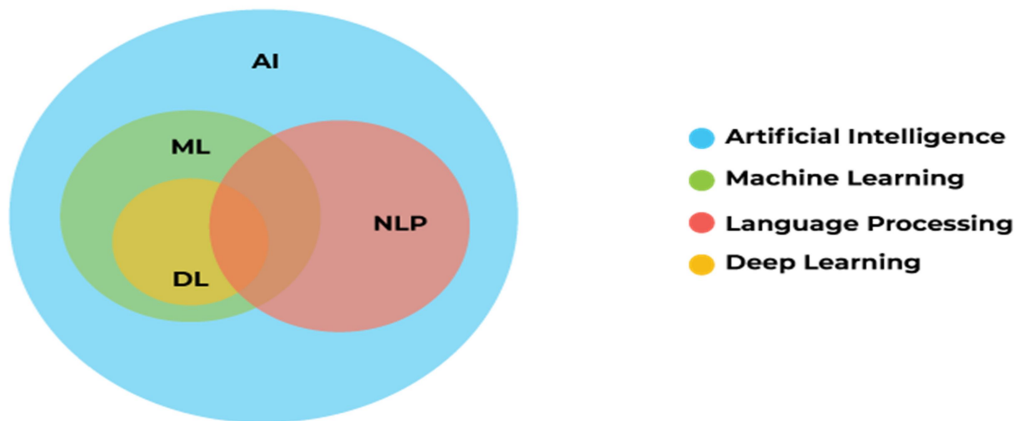


Figure 6 Natural Language Processing (NLP)

3.4.2 Natural Language Toolkit

The NLTK module is a massive tool kit, aimed at helping you with the entire Natural Language Processing (NLP) methodology. NLTK will aid you with everything from splitting sentences from paragraphs, splitting up words, recognizing the part of speech of those words, highlighting the main subjects, and then even with helping your machine to understand what the text is all about. In this series, we're going to tackle the field of opinion mining, or sentiment analysis.

3.4.3 Gensim

It is another NLP library for Python primarily developed for topic modeling. However, it now supports a variety of other NLP tasks such as converting words to vectors (word2vec), document to vectors (doc2vec), finding text similarity, and text summarization. It lets you handle large text files without having to load the entire file in memory.

3.4.4 Google Search

Google Search (or Google Web Search) is a web search engine owned by Google Inc. Google Search, is the most-used search engine on the World Wide Web, receiving several hundred million queries each day through its various services. There are several steps we need to go through to gather data from Internet to check if a sentence has been plagiarized. The first one is to use a search engine. Google is probably the fastest and trustworthy search engine there is

nowadays. We assume that if Google cannot find a sentence at all, then it doesn't come from Internet and therefore, this sentence hasn't been plagiarized

3.4.5 Web Scraping

Web scraping is data scraping used for extracting data from websites. Web scraping software may access the World Wide Web directly using the Hypertext Transfer Protocol, or through a web browser.

Web scraping a web page involves fetching it and extracting from it. Fetching is the downloading of a page (which a browser does when a user views a page). Therefore, web crawling is a main component of web scraping, to fetch pages for later processing. Once fetched, then extraction can take place. The content of a page may be parsed, searched, reformatted, its data copied into a spreadsheet, and so on. Web scrapers typically take something out of a page, to make use of it for another purpose somewhere else.

Web pages are built using text-based mark-up languages (HTML and XHTML), and frequently contain a wealth of useful data in text form. However, most web pages are designed for human end-users and not for ease of automated use. As a result, specialized tools and software have been developed to facilitate the scraping of web pages.

3.4.6 Bing Web Search API

The Bing Web Search API is a RESTful service that provides instant answers to user queries. Search results are easily configured to include web pages, images, videos, news, translations, and more. Bing Web Search provides the results as JSON based on search relevance and your Bing Web Search subscriptions.

This API is optimal for applications that need access to all content that is relevant to a user's search query. It requires only a specific type of result; consider using the Bing Image Search API, Bing Video Search API, or Bing News Search API.

3.4.7 TF-IDF

TF-IDF stands for term frequency inverse document frequency used in an information retrieval and text mining. Each word or term that occurs in the text has its respective TF and IDF score. The product of TF and IDF scores of a term is called the TF*IDF weight of that term. 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.

3.4.8 How to compute TF*IDF

TF: Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more often in long documents than shorter ones. Thus, the term frequency is often divided by the document length (i.e. the total number of terms in the document) as a way of normalization:

$$TF(t) = (\text{Number of times term } t \text{ appears in a document}) / (\text{Total number of terms in the document}).$$

IDF: Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

$$IDF(t) = \log_e(\text{Total number of documents} / \text{Number of documents with term } t \text{ in it}).$$

See below for a simple example.

Example: Consider a document containing 100 words wherein the word *cat* appears 3 times. The term frequency (i.e., TF) for *cat* is then $(3 / 100) = 0.03$. Now, assume we have 10 million documents and the word *cat* appears in one thousands of these. Then, the inverse document frequency (i.e., IDF) is calculated as $\log(10,000,000 / 1,000) = 4$. Thus, the TF*IDF weight is the product of these quantities: $0.03 * 4 = 0.12$.

3.4.9 Python

Python is an easy, interpreted, object-oriented, high-level programming language with dynamic semantics. It is a general-purpose coding language, unlike HTML, CSS, and JavaScript, it can be used for types of programming and software development besides web development. That includes back end development, software development, data science and writing system scripts among other things.

Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.

The reason why we choose python programming language for our system development is that it consist many natural language libraries which we have used in this project and can easily integrate with machine learning to extend and implement it for the future work.

3.4.10 Flask

Flask is a back-end python web framework built with a small core and easy to extend philosophy. Flask provides you with tools, libraries and technologies that allow you to build a web application. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. We have been used Flask framework for the entire work by installing necessary libraries using command 'pip install'

3.4.11 Java-script (js)

JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage the user.

3.4.12 Html

Hyper Text Mark-up Language (HTML) is a mark-up language used to structure a web page and its content. HTML is heavily used for creating pages by integrating either scripting or in other programming language frame-works like Flask by using template engine.

3.4.13 Bootstrap

It is a front-end framework which is used to create user interfaces in web applications. It provides css, js and other tools that help to create the required interface. Bootstrap is friendly and can be used in any mark-up or programming languages (Flask in this case).

CHAPTER-4

RESULTS AND DISCUSSIONS

4.1 Result

This is how our system home page looks like it is a web based application. When user opens the system the main page will display as shown in the following below.

The screenshot displays the home page of a plagiarism detection system. At the top, the header includes the text "Addis Ababa Science And Technology University" and a logo. Below the header, a navigation bar contains the text "STUDENT'S ASSIGNMENT CHECKING SITE" and "PLAGIARISM DETECTION SYSTEM". The main content area is divided into three sections. On the left, there is a large text input field with the placeholder text "Place some text here". Below this field, there is a "Choose File" button and the text "No file chosen". Further down, it says "word limit:1000" and has two buttons: "CHECK" and "REFRESH". In the center, there are two circular progress indicators, both showing "0%". On the right, there is a large empty rectangular area. At the bottom of the page, there is a footer with the text "2B@aastu.edu.et" and a Windows watermark that says "Activate Windows Go to Settings to activate Windows."

Figure 7 Home Page

Then after, the system allows the users either to upload or copy-paste the content that needed be checked for plagiarism in the provided text area.

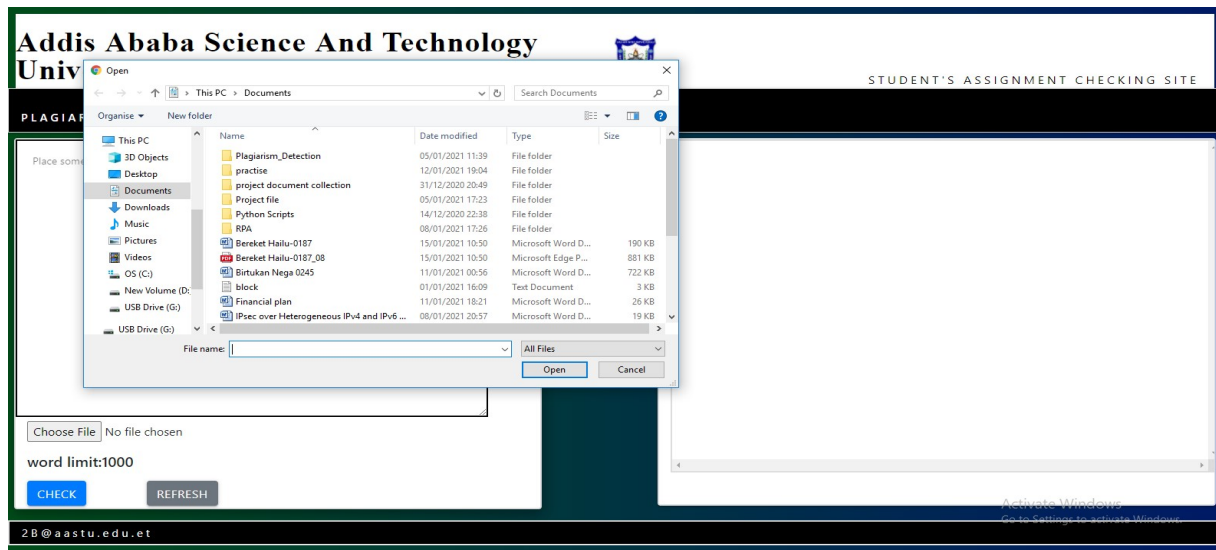


Figure 8 Input Receivers

If a user uploads a file in different file formats like (.txt .pdf .docx) it immediately extracts the content of file and posts it in the given provided text-area. After clicking the check button it will check the similarity for each similar URL and show the maximum and minimum output result in circular progress bar.

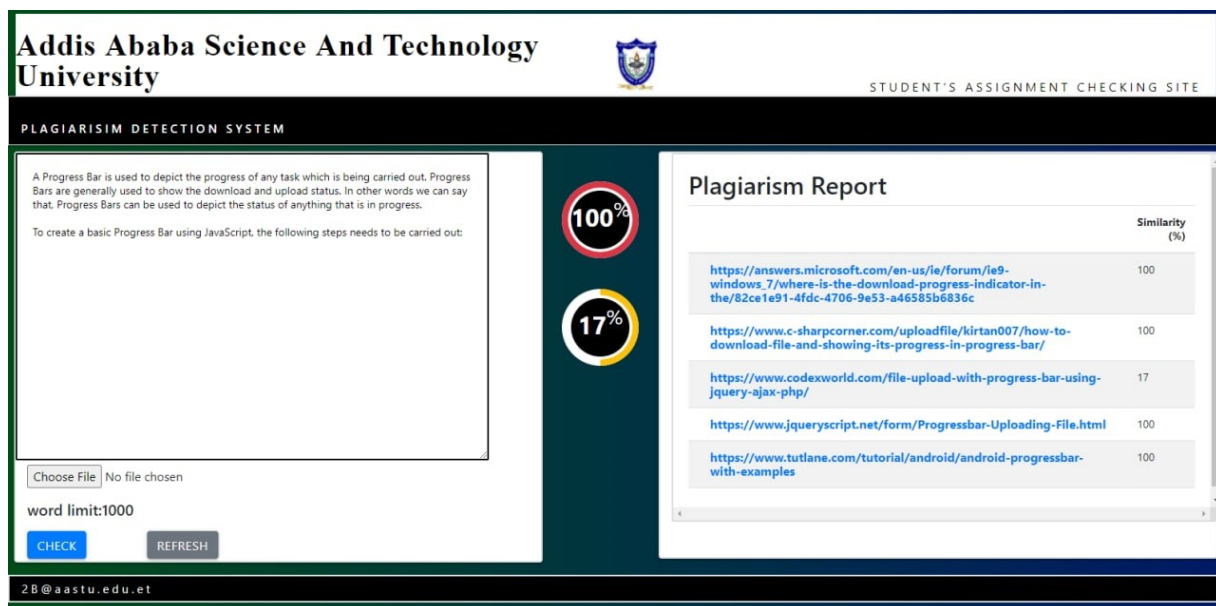


Figure 9 Output

CHAPTER-5

CONCLUSIONS AND SCOPE FOR FUTURE WORK

5.1 Conclusions

Plagiarism of text has become a common occurrence today with difficulties to detect forms such as paraphrasing and summarizing being frequently practiced. There is a need to design effective mechanisms for automated plagiarism detection. In the proposed system we have tried to overcome the current problems by using paraphrase (intelligent-based) recognition approach based on NLTK techniques for preprocessing the text for cleaning purpose and TF-IDF to convert text format in to machine readable binary format and we have also used generate similarity (GENSIM) in order to compute for similarities. In this way we can detect and prevent plagiarism in the papers submitted in order to promote academic integrity and honesty using an intelligent system. So that we protect the patent of the original content creators and also helps to assure the contents submitted are their own work.

5.2 Scope for Future Work

For the future this system can be improved using different techniques to gain more accurate result by applying modern machine learning algorithms.

REFERENCE

- [1] A. Chitra and A. Rajkumar: Plagiarism detection using paraphrase recognizer
- [2] Alsallal, M. (2016) A Machine Learning Approach for Plagiarism Detection. Unpublished PhD Thesis. Coventry: Coventry University
- [3] Oxford English Dictionary :qtd. in Lands(1999).
- [4] Random House Compact Unabridged Dictionary, 1995; Oxford English Dictionary, 1999
- [5] Rune BorgeKalleberg: Towards Detecting Textual Plagiarism Using Machine Learning Methods.
- [6] <http://wpacouncil.org/positions/WPAplagiarism.pdf>[7] <https://www.ukessays.com/essays/computer-science/proposed-system-plagiarism-detection-9544.php>
- [8] Potthast,M., Stein, B.,. Ceden A.B, and Rosso, P. An evaluation framework for plagiarism detection. Proc. of 23rd Int. Conf. on Computational Linguistics, COLING , Beijing, China (2010)
- [9] Alzahrani, S. M., Salim,N., and Abraham, A. Understanding plagiarism linguistic patterns, textual features, and detection methods. IEEE transactions on systems, man, and cybernetics part c: application and reviews, 42 (2) (2012).
- [10] Torrejon, R. D. A., and Ramos, M. J. M. Text Alignment Module in CoReMo 2.1 Plagiarism Detector- lab report for PAN at CLEF 2013. Proc. of 5th Int.Workshop PAN-13, Valencia, Spain (2013).
- [11] Alzahrani,S.M., and N. Salim. Fuzzy semantic-based string similarity for extrinsic plagiarism detection- lab report for PAN at CLEF 2010. Proc. of 2nd Int. Workshop PAN-10, Padua, Italy(2010).
- [12] Ceska, Z. Plagiarism detection based on singular value decomposition. Advances in Natural Language Processing, Springer Berlin Heidelberg Lecture Notes in Computer Science, 52(21),108–119(2009).
- [13]<https://elearningindustry.com/top-10-free-plagiarism-detection-tools-for-teachers>
- [14] Redfern, K. and Barnwell, N. (2009) ‘Cultural Differences in Attitudes toward Plagiarism in Undergraduate Business Students’: An Empirical Investigation. Unpublished Thesis: University of Technology Sydney Australia
- [15] Roig, M. (2001) ‘Plagiarism and Paraphrasing Criteria of College and University Professors’. Ethics & Behaviour 11 (3), 307 -323

APPENDIX – I

PROJECT CODE OF IMPORTANT STEPS

```
1  import nltk
2  import gensim
3  import websearch
4  from difflib import SequenceMatcher
5  import pandas as pd
6  import numpy as np
7  from nltk.corpus import stopwords
8  from nltk.stem import PorterStemmer
9  from nltk.tokenize import word_tokenize, sent_tokenize
10 from sklearn.feature_extraction.text import TfidfVectorizer
11 from sklearn.metrics.pairwise import cosine_similarity
12
13 pst = PorterStemmer()
14 nltk.download('stopwords')
15 nltk.download('punkt')
16 stop_words = set(nltk.corpus.stopwords.words('english'))
17 import re
18 punctuation=re.compile(r'[-.?!,,:;()|0-9]')
19
20 def purifyText(string):
21     text1 = re.sub(r'\[[0-9]*\]', ' ', string)
22     text1 = re.sub(r'\s+', ' ', text1)
23     text1 = re.sub(r'\d', ' ', text1)
24     text1 = re.sub(r'\s+', ' ', text1)
25     words = nltk.word_tokenize(text1)
26     return (" ".join([word for word in words if word not in stop_words]))
27
28 def webVerify(string, results_per_sentence):
29     sentences = nltk.sent_tokenize(string)
30     matching_sites = [] #list of url come from websearch/searchBing
31     # for url in websearch.searchBing(query=string, num=results_per_sentence):
32     #     matching_sites.append(url)
33     for sentence in sentences:
34         for url in websearch.searchBing(query = sentence, num = results_per_sentence):
35             matching_sites.append(url)
36     return (list(set(matching_sites)))
37
```

```

38 def similarity(str1, str2):
39     file_docs = []
40     doc1 = re.sub(r'\\[0-9]*\\', ' ', str1)
41     doc1 = re.sub(r'\\s+', ' ', doc1)
42     doc1 = re.sub(r'\\d', ' ', doc1)
43     doc1 = re.sub(r'\\s+', ' ', doc1)
44     tokens = sent_tokenize(doc1)
45
46     #add tokenized sentences in array/list
47     for line in tokens:
48         file_docs.append(line)
49
50     #tokenize words and create dictionary
51     gen_docs = [[pst.stem(w.lower()) for w in word_tokenize(text)]
52                 for text in file_docs]
53
54     #create a Dictionary object that maps each word to a unique id.
55     dictionary = gensim.corpora.Dictionary(gen_docs)
56
57     #create a bag of words corpus and pass the tokenized list of words to the Dictionary.doc2bow()
58     corpus = [dictionary.doc2bow(gen_doc) for gen_doc in gen_docs]
59
60     #TFIDF down weights tokens (words) that appears frequently across documents.
61     tf_idf = gensim.models.TfidfModel(corpus)
62
63     #Creating similarity measure object
64     sims = gensim.similarities.Similarity('C:\\Users\\User\\Documents\\Plagiarism_Detection', tf_idf[corpus],
65                                           num_features=len(dictionary))
66
67     file2_docs = []
68     doc2 = re.sub(r'\\[0-9]*\\', ' ', str2)
69     doc2 = re.sub(r'\\s+', ' ', doc2)
70     doc2 = re.sub(r'\\d', ' ', doc2)
71     doc2 = re.sub(r'\\s+', ' ', doc2)
72     tokens = sent_tokenize(doc2)
73
74     for line in tokens:
75         # word=punctuation.sub("",line)
76         file2_docs.append(line)
77     avg_sims = []
78     for line in file2_docs:
79         query_doc = [w.lower() for w in word_tokenize(line)]
80         query_doc_bow = dictionary.doc2bow(query_doc)
81         query_doc_tf_idf = tf_idf[query_doc_bow]
82
83         sum_of_sims =(np.sum(sims[query_doc_tf_idf], dtype=np.float32))

```

```

for line in tokens:
    # word=punctuation.sub("",line)
    file2_docs.append(line)
avg_sims = []
for line in file2_docs:
    query_doc = [w.lower() for w in word_tokenize(line)]
    query_doc_bow = dictionary.doc2bow(query_doc)
    query_doc_tf_idf = tf_idf[query_doc_bow]

    sum_of_sims =(np.sum(sims[query_doc_tf_idf], dtype=np.float32))
    # calculate average of similarity for each query doc
    avg = sum_of_sims / len(file_docs)
    # print average of similarity for each query doc
    print(f'avg: {sum_of_sims / len(file_docs)}')
    # add average values into array
    avg_sims.append(avg)

# calculate total average
total_avg = np.sum(avg_sims, dtype=np.float)
# round the value and multiply by 100 to format it as percentage
percentage_of_similarity = round(float(total_avg) * 100)
# if percentage is greater than 100
# that means documents are almost same
if percentage_of_similarity >= 100:
    percentage_of_similarity = 100
return percentage_of_similarity
# return (SequenceMatcher(None,str1,str2).ratio())*100

def reports(text):

    matching_sites = webVerify(purifyText(text), 5)
    matches = {}

    for i in range(len(matching_sites)):
        matches[matching_sites[i]] = similarity(text, websearch.extractText(matching_sites[i]))

    matches = {k: v for k, v in sorted(matches.items(), key=lambda item: item[1], reverse=True)}

    return matches

def Table(dictionary):

    df = pd.DataFrame({'Similarity (%)': dictionary})
    # tot_avg = np.sum(v)
    #df = df.fillna(' ').T
    #df = df.transpose()
    html = df.to_html(classes='table table-striped', header="true", render_links=True, bold_rows=True, justify='right', notebook=True)
    # html = html.style.set_properties({'width': '300px'})
    return html

```