AI-POWERED PLAGIARISM DETECTION: LEVERAGING FORENSIC LINGUISTICS AND NATURAL LANGUAGE PROCESSING

Article *in* FUDMA Journal of Sciences · September 2021

DOI: 10.33003/fjs-2021-0503-700

3 authors, including:

Anthony Nwohiri
University of Lagos
18 PUBLICATIONS **25** CITATIONS
SEE PROFILE

Olasupo Ajayi
University of the Western Cape
64 PUBLICATIONS **268** CITATIONS
SEE PROFILE

FUDMA Journal of Sciences (FJS)
ISSN online: 2616-1370
ISSN print: 2645 - 2944
Vol. 5 No. 3, September, 2021, pp 207 - 218
DOI: https://doi.org/10.33003/fjs-2021-0503-700

AI-POWERED PLAGIARISM DETECTION: LEVERAGING FORENSIC LINGUISTICS AND NATURAL LANGUAGE PROCESSING

[1*]Nwohiri, A.M., [2]Joda, O.O., [3]Ajayi, O.O.

[1]Department of Computer Sciences, University of Lagos, Lagos, Nigeria
[2]Palantir Technologies, London, United Kingdom
[3]Department of Computer Sciences, University of the Western Cape, Cape Town, South Africa

*Corresponding author's Email: anwohiri@unilag.edu.ng

ABSTRACT
Plagiarism of material from the Internet is nothing new to academia and it is particularly rampant. This challenge can range from borrowing a particularly apt phrase without attribution, to paraphrasing someone else's original idea without citation, to wholesale contract cheating. Plagiarized content can infringe on copyright laws and could incur hefty fines on publishers and authors. Unintentional plagiarism mostly occurs due to inaccurate citation. Most plagiarism checkers ignore this fact. Moreover, plagiarizers are increasingly becoming negatively "smarter". All these necessitate a plagiarism detector that would efficiently handle the challenges. Several plagiarism detectors have been developed but each with its own peculiar limitations. This paper aims at developing an AI-driven plagiarism detector that can crawl the web to index articles and documents, generate similarity score between two local documents, train users on how to properly format in-text citations, identify source code plagiarism and use natural language processing and forensic linguistics to properly analyse plagiarism index.

Keywords: Academia, Forensic linguistics, Natural language processing, Plagiarism, Plagiarism checker,

INTRODUCTION

Plagiarism has existed ever since the dawn of human creativity. The Centre for Academic Integrity disclosed that as at two decades ago, about 1 in every 10 students admitted to having been engaged in some form of plagiarism. By the early 2000s, the number had risen to 4 in 10 (ICAI, 2021). A survey conducted by Rutgers University professor Donald McCabe over the course of three years (2002-2005) reported that 38% of students admitted to "paraphrasing/copying few sentences from written source without footnoting it (McCabe, 2005). This alarming increase can be attributed to the ever-easier access to technology. Internet access is making plagiarism easier, hence educational institutions today need to educate students and teachers on the various kinds of plagiarism and how they can be avoided.

The word plagiarism first came into use in the early 17th Century and has its root in the Latin term plagiarius, meaning kidnapper or plunderer, which in itself is rooted in the Greek word plagion, for oblique or non-direct (Ison, 2017). Plagiarism is the use or close imitation of the language and thoughts of another author and the representation of them as one's own original work (RHIG, 2021). In academia, this can range from borrowing without attribution a particularly apt phrase, to paraphrasing someone else's original idea without citation, to wholesale contract cheating (Pennycook, 1996). When you take the writings of another person and pass them off as yours, you are plagiarizing. This is closely attributable to forgery and piracy (Britannica, 2021). Plagiarism has always been a major problem in our society, especially now that people can simply "copy and paste" from the Internet. Easy access to information has made stealing information, research and solutions from other authors much easier. This is not just unethical, but it could affect the integrity and/or profits of the original author(s) and the source article.

According to Szuchman (2010), students are often surprised to learn of the variety of behaviours that constitute plagiarism. However, whether intentional or not, the penalties can be severe. With plagiarism, as with criminal law, ignorance is no excuse. Unintentional plagiarism occurs when a student or researcher does not do enough research into the topic to see if his ideas already exist in published articles. It can also be as a result of forgetting to cite their sources, or misquotes their sources, or unintentionally paraphrases a source by using similar words, groups of words, and/or sentence structure without attribution (DeLong, 2012).

To mitigating plagiarism, several plagiarism checkers have been developed over the years (PlagAware, 2021; Checkforplagiarism, 2021; KIT, 2021; Turnitin, 2021; Blackboard, 2021); each with its peculiar limitations. Moreover, plagiarizers are becoming negatively "smarter" and can outsmart these systems. Rearranging and paraphrasing content could successfully trick some of these plagiarism checkers. In addition, there are numerous free online paraphrasing tools powered by Artificial Intelligence (AI), that are able to evade many plagiarism detectors. These tools modify "stolen contents" to such a degree as to evade even the most advanced copy content scanning software (Kumar, 2021).

An AI-enhanced plagiarism checkers can therefore be a viable solution to this plagiarism menace. The contributions of this paper is thus to develop such a plagiarism checker that

employs AI technologies, specifically forensic linguistics and Natural Language Processing (NLP), to detect plagiarism. NLP is a form of computational linguistics which uses methods borrowed from computer science, artificial intelligence, linguistics and data science, to enable computers understand, interpret and manipulate human language (Dataman, 2021). By leveraging on these technologies, our proposed plagiarism does not just compare texts at face value but is able to understand the textual content to a certain degree and decide if an idea was plagiarized or not.

The rest of this paper is organized as follows: section 2 reviews a number of existing plagiarism checkers, while the various methods of detecting plagiarism are discussed in section 3. Our methodology and system designs are presented in section 4, while system implementation and experimental results are detailed in section 5. Section 6 concludes the paper.

REVIEW OF RELATED LITERATURE

There have been several attempts at creating plagiarism checkers, some of which are reviewed in this section.

In Vani and Gupta (2016), the authors argued that intelligent techniques for detection of high obfuscations are still in their infancy and most of the available online, standalone and web-based tools are unable to detect complex manipulations. Foltynek et al. (2020) evaluated the performance of 15 plagiarism checkers from both the coverage and usability perspectives. Texts from 8 different languages including Czech, English, German, Italian, Latvian, Slovak, Spanish, and Turkish; with Wikipedia, online publications and academic theses being the primary sources. It was concluded that better results were obtained for major languages than minor ones, the source of the document significantly influenced the performance of these checkers, and plagiarism from single sources were more difficult to detect than those from multiple sources. Pertile et al. (2016) analysed plagiarism checkers based on their ability to detect content (including paraphrasing), citations and structural similarities. In a study on factors causing plagiarism in student papers, Sulaiman (2018) identified lack of understanding of how to cite and write references, limited access to referrals and student attitudes as the primary causes. Oladeji *et al.* (2020) proposed a plagiarism checker for academic theses from Nigerian universities. The proposed model crawls a network of local institutional repositories (IRs) to check for pre-existing theses or dissertations. A fundamental requirement to their proposed model, is a well-established network of constantly updated IRs; without there would be no "source" content against which a plagiarised text is checked.

3. REVIEW OF EXISTING PLAGIARISM CHECKER

There are numerous plagiarism checkers available commercially. Some of which are available include: PlagAware (PlagAware, 2021), CheckForPlagiarism.net (Checkforplagiarism, 2021), JPlag (KIT, 2021), Turnitin (Turnitin, 2021), SafeAssign (SafeAssign, 2021).

PlagAware is a search engine tool which operates in two modes. In the first mode - plagiarism assessment, it assumes that a published document is a potential plagiarism of a resource on the web, the document is then analysed to detect the extent of plagiarism content therein. In the second mode - text monitoring, an active monitor is placed on a given resource, document or web page. This monitor assumes the article to be an original and actively scans all places and new documents for plagiarisms. A real time update is given to the author when a plagiarism of document is found (PlagAware, 2021).

CheckForPlagiarism.net makes use of document source comparison methods such as finger- printing. A numeric value (fingerprint) is assigned to each document after being processed, such that documents with similar contents have their numerical values close to each other. This makes it easy to find similar documents among billions of files. Plagiarism check is simply a matter of selecting the documents with values close to the submission rather than an in-depth analysis on suspected documents. JPlag is used for source code comparison. It can detect similarities between multiple source code files. To ensure it detects software plagiarism, it does not merely follow document comparison principles because a plagiarism of a source code may not have similar textual content, JPlag is aware of how programs compile and keeps a record of programming language syntax. This means it would still be able to detect plagiarism even after common attempts to disguise similarities, such as renaming variables or function names. JPlag currently supports Java, C, C and C++. It can also process natural language text. Turnitin is an Internet-based plagiarism detection service. Documents to be checked for plagiarism are uploaded to remote servers from where the system creates fingerprints of the documents and stores them. Turnitin creates originality report by scanning a submitted document against contents in its database. Turnitin has a robust database, which includes over 45 billion web pages, 130 million journal articles and 150 million student papers. This large database size makes Turnitin able to detect a wide variety of plagiarized document content. SafeAssign, like Turnitin is also a web-based plagiarism service, which searches the Internet and several third-party databases. It is also capable of scanning and indexing zipped and password protected documents. Each document has their fingerprint stored in different databases, a different database is created for all institutions, therefore submissions are never compared with documents from another institutions. This unique feature help prevent privacy policies violations and copyright laws.

4. CATEGORIES OF PLAGIARISM DETECTORS

Plagiarism detection can be grouped into three main categories: document source comparison, manual search of characteristic phrases, and stylometry.

• *Document source comparison:* Here a document is compared against a set of documents to see exactly how similar that document is to any of the documents in the corpus. To compare these documents, moderately sized string or fingerprints are compared with an already processed index table to detect similarities between documents. The fingerprint generation (Hoad and Zobel, 2003; Rezaeian and Novikova, 2017; Shivakumar and Garcia-Molina, 1999) process of a document and its comparison with another document is shown in Figure 1. It works as follows: each document is divided into a set of continuous pieces of tokens; a Hash function is performed on the pieces and fingerprints are created; some fingerprints are selected based on the selection strategy; the fingerprints of two documents are compared with each other such that if two documents have the same fingerprints at least within the threshold, the system considers them as identical.

It is important to note that the specifics of the fingerprint of a document are not constant. Different researchers and organizations use different (modified) mechanisms for creating fingerprints in order to improve similarities detection processes. In essence there are no definitive mechanisms that can optimally always detect similarities in all documents.

• *Manually searching of characteristic phrases:* This process considers characteristic phrases within a document, i.e., phrases or words that are uncommon and unique to that document (characteristics of the document). The phrases are then fed into a search engine such as Google. The resulting articles generated are analysed against the source document to analyse the degree of plagiarism. This approach is clearly labour intensive as it would involve making several searches on each document; hence it makes sense to automate the process. SNITCH was developed to automate this process. A mixture of word lengths, word frequencies and phrase length are used to compute these characteristic phrases (Niezgoda and Way, 2006).

• *Stylometry:* Intrinsic plagiarism detection is when an author plagiarizes their past work without proper references. Stylometry is statistical analysis of literary style. It involves doing a style analysis of a document or performing style comparisons of documents written by the same author. It is based on a theory that each person has a unique style of writing hence, able to easily detect when an author recycles any of his ideas from previous articles. If a document has a lot of similar substrings with another, it is marked as a plagiarism. In academic environments there are usually large amounts of documents to be analysed, thus this method of detection might not be feasible but could be used as a secondary layer of detection (Meyer zu Eissen and Stein, 2006).

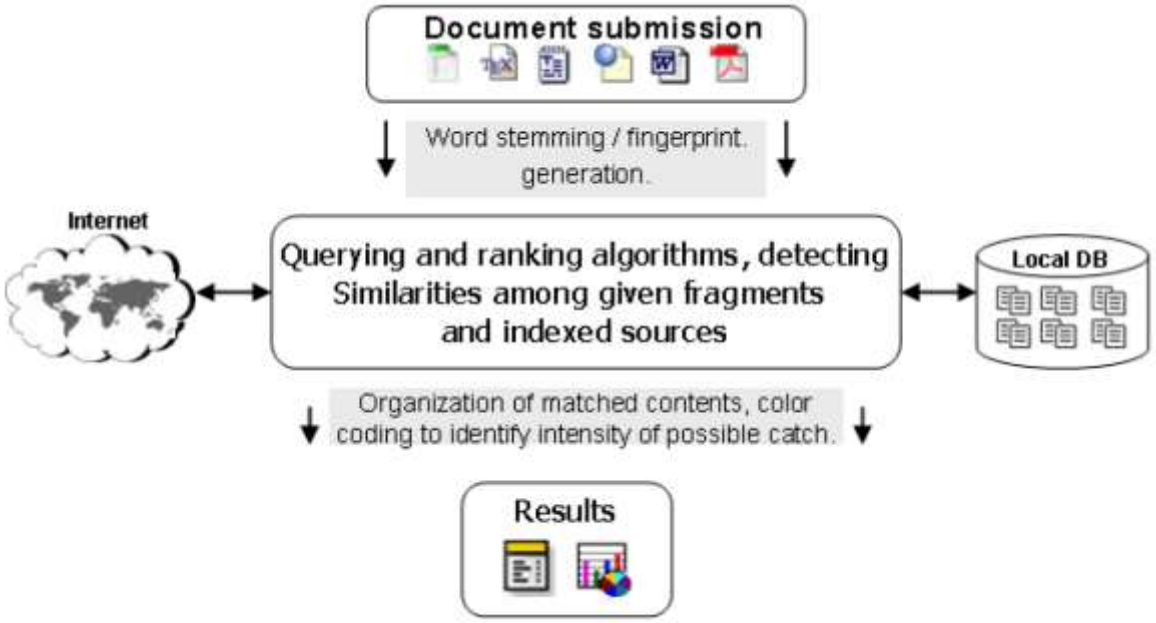

Figure 1: Document source comparison in plagiarism detection (Rezaeian and Novikova, 2017)

PLAGIARISM TECHNIQUES

Most of the tools and services described above are very excellent in detecting verbatim plagiarism. However, word-based fingerprinting, which most of these tools are based on, fails when the plagiarism or similarities in the documents are not word based. Paraphrasing could also be done in such a way that 2 sentences mean the exact same thing but do not have any words in common.

Although there are no existing plagiarism detection systems using semantic similarities or a concept-oriented search as opposed to word-based indexing, more sophisticated methods of detection are already in use in much smaller scales e.g., checking if a question asked is similar to any question already available in a list of FAQs. In these cases, it is possible to compare certain texts with other texts to detect semantic similarities. However, this is not feasible when a submission is to be compared to billions of documents. When dealing with a large document corpus, documents need to be assigned a value such that similar documents have closer values, this allows picking out similar document from a large corpus without per-document based comparison. Recent research in this direction suggests that the best approach to checking if a similarity exists is to determine semantic equivalence, analyse a document and assign it a value such that any two documents that are semantically similar will have their values close to each other. "*To actually prove that two pieces of text are semantically equivalent one would require a complete understanding of natural language, something still quite elusive. However, a compromise can be considered: rather than allowing a full natural language we restrict our attention to a simplified grammar and to a particular domain for which an ontology (semantic network) is developed. Clearly, sufficiently restricting syntactic possibilities and terms to be used will allow one to actually prove the equivalence of pieces of text.*" (Heinrich and Maurer, 2000).

Table 1: Word-frequency in text and complete vocabulary

COMPLETE VOCABULARY	PHRASE A	PHRASE B
The	The: 2	The: 1
Boy	Boy: 1	
Annie		Annie: 1
Owns	Owns: 1	
Likes		Likes: 1
To		To: 1
Red	Red: 1	
Drive		Drive: 1
Car	Car: 1	
Truck		Truck: 1

A vector space model is a very common approach to detecting similarities in documents. These models determine how close/similar two texts are by converting them into vectors and calculating the distance (or cosine similarity) between them. Cosine similarity is calculated as the cosine of the angle between 2 vectors. As an example: given two phrases; Phrase A: "The girl owns the green car" and Phrase B: "John likes to drive the truck". Each phrase can be represented in a word-frequency table shown on Table 1.

To represent phrase A and phrase B as vectors that are based on the complete vocabulary defined above, then: Phrase A = 2,1,0,1,0,0,1,0,1,0, Phrase B = 1,0,1,0,1,1,0,1,0,1 These vectors are compared with the vector of a new document to detect similarities. For example, if we have a Phrase C given as: "The green truck", the vector representation, based on the same vocabulary is given as: Phrase C = 1,0,0,0,0,0,1,0,0,1.

The cosine similarity of any 2 documents A and B can be calculated by substituting their vectors into this equation:

$$\frac{V(A) \bullet V(B)}{|V(A)||V(B)|}$$

where V(X)is the vector representation of text X.

With this equation the similarity between Phrase A and Phrase C can be calculated using their respective vectors. After calculations, Phrase A and Phrase C give a similarity measure of 0.61, while the similarity between Phrase B and Phrase C is 0.47. Phrases A and C have a relatively high similarity score, therefore, Phrase C can be said to be a plagiarism of Phrase A.

Stop words are words that are repeated so often and occur in most documents. They include words such as 'a', 'be', 'the', 'of', 'is' ... etc. The words are usually removed before creating the vectors to give more accurate results. Furthermore, words within sentences are weighted using tf-idf (term frequency–inverse document frequency) weights. This ensures that rare words have a higher weight, because of their uniqueness, a repetition is almost likely plagiarism. Moreover, vectors for large texts would take a lot of space. To minimize this, the complete vocabulary can be limited to words that are relevant to the subject in question. There are services available that analyse the conceptual content of a document, such as Latent Semantic Analysis (LSA) and Normalized Word Vectors (NWV) (Dreher and Williams, 2006).

Comparing all these software shows that there is no "perfect" plagiarism detector yet. Since they all use different approaches which have their different advantages and disadvantages. For example, the fingerprint approach used by CheckForPlagiarism.net will perform well when checking if a document plagiarizes another but will perform poorly if a plagiarizer copies chunks of content from several different documents. Similarly, the similarity index used by Turnitin is useful when showing what portions of the document was plagiarized but might not perform well after an online paraphrasing tool has scrambled the text (Ali et al., 2011). JPlag can handle some programming languages but not all and cannot handle textual plagiarism. There are recommendations that teachers should educate students about plagiarism and its impact, copyright, citation and ownership. However, most of these tools do not provide such information on them; hence, there is a need to identify and correct accidental plagiarism. Current plagiarism detection tools have a few common shortcomings, including:

- If Plagiarist makes an active attempt to beat the plagiarism detector by using synonymizing or paraphrasing tool, the syntactic version of the output, will be different, even though the content means the same thing.
- If the content that was plagiarized is not online or available in any of the databases that the plagiarism checker has access to.
- When plagiarism crosses language boundaries. Cross-language plagiarism will continue to be a challenge in the immediate future, but it is not the focus of this paper. Offline or unavailable

documents are becoming less of a challenge with organizations such as Google plan to scan most of the texts in many standard libraries. Furthermore, the fact that most articles and books being published in recent times are also made available in soft or digital format. Web crawlers would be able to obtain and index them. Finally, regarding paraphrasing, a good implementation of natural language processing will enable semantic analysis of sentences to determine similar rather than relying on only words.

These comparisons show that though these tools are very good in detecting matching text between documents, even advanced plagiarism detectors cannot detect plagiarism as good as humans do. They have several drawbacks that would not fool a trained human eye. The human brain is the ideal plagiarism detection tool, for both textual and source code detection, subconsciously using statistics and semantics to accurately assess document plagiarism.

ANALYSIS AND METHODOLOGY

In this section, the methodology used in developing our AI-enhanced plagiarism checker is discussed. Our system accepts a document or a group of documents as input and generates a plagiarism report as well as improvement suggestions. The system is capable of handling large document corpus, textual plagiarism and source code plagiarism with a high degree of efficiency.

The problem of plagiarism has been tackled using software in the past. However, these solutions have become less effective with swift evolution of technology. There is a need to take another look at the plagiarism problem through modern lenses. A new system for checking plagiarism has to be developed to tackle new plagiarism problems that are emerging due to technological advances.

a. System Flow

A high level information flow amongst entities in the proposed system is shown in Figure 2.

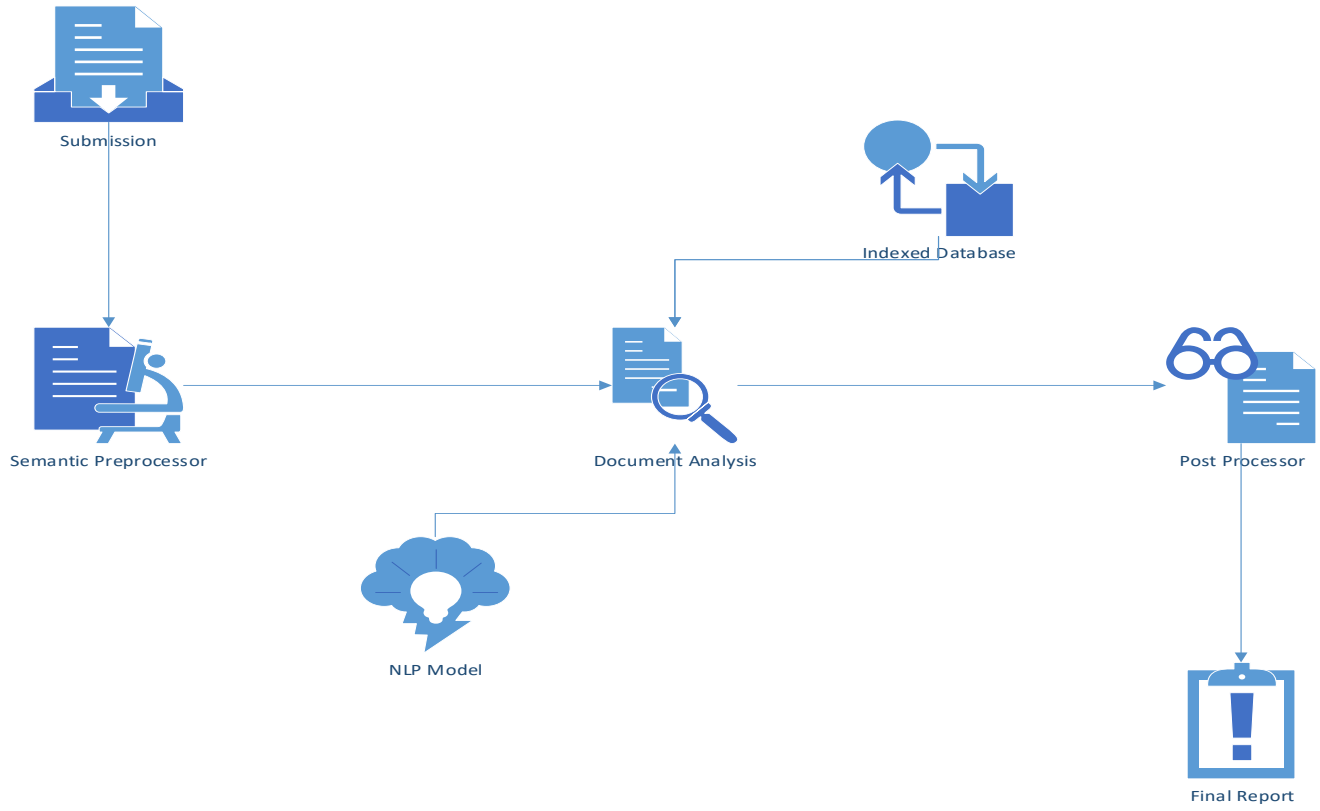

Figure 2: System flow for the proposed system

This is the flow to generate a report from the submission. The system is meant to take in a submission as a document, pass it to the pre-processor which extracts useful information from the document and converts it to the data structure that will be used internally. It is then analyzed using a Natural Language Processor (NLP) model against documents already stored in the database. The result is subsequently cleaned up in the post-processor and an output in form of a comprehensive report is generated. The individual component of the system are described as follows:

- The document entity represents any text, submission, article, paper or project that is either already stored in the Document Corpus (Document DB) or is being passed in to be analysed against documents already stored. Any document input to be analysed will also be stored afterwards, and used to analyse other future documents.
- Report is the output from any plagiarism detection process. After the system has analysed a submitted document, identified similar documents, compared with the submission and identified areas of

plagiarism, taking into account references and quotes. All this information is gathered and presented in an explanatory fashion on a report to be returned to the user.

- Semantic Pre-processor cleans up the document and ensures it is ready to be acted on. All stop words are removed, words are stemmed and mapped to a root word. This ensures better matching during search, and removes lots of unnecessary information that would have been stored otherwise.
- Inverted Index DB is a Map that has Documents as values, not keys. What this means is documents are not perused through, because that would take too much time. Instead we calculate some other information, the document fingerprint, that would be easy to locate or collect, after searching for close/similar fingerprints. the Inverted Index can return the associated document(s).
- NLP model would be generated for semantic fingerprinting, such that documents which are conceptually or semantically similar would have values close to each other. This information is stored in the inverted index. A new document to be assessed will pass through this model, and the result will be searched through the database for close or similar matches.
- Language Parser is used for source code plagiarism. The application must be able to understand how languages work, for every language to be supported there must be a parser that can parse the source code to a form that can be analysed for similarities and plagiarism.
- Post-processor This is where all the information gathered by the previous steps is accu- mulated and built into a comprehensive report for the user.

Design Architecture

The system follows a Model-View-Controller (MVC) architecture. MVC is a design pattern used for developing interfaces. It divides the program logic into three interconnected elements –

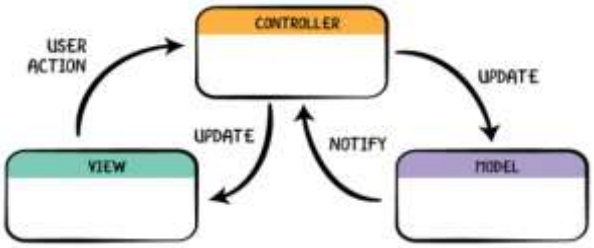

Figure 3: Illustration of MVC Architecture (Marcos, 2019)

Model, View and Controller – to separate internal representations of information from the ways information is presented to and accepted from the user.

The Model is the data that is processed and/or stored by the system. In our system the Model is the inverted index database which holds information of all analysed documents. The View is often times the user interface and provides a means of displaying data in an application. In our system, the View is a front-end web interface used by both lecturers and students. The Controller is the process that handles the inputs, processing and outputs. In our system, the application back-end developed in Python serves this purpose. It handles input received from the users (and information from the Model) to generate output for the View (and update the Model). Figure 3 shows an illustration of the MVC architecture.

System Workflow

Figures 4 and 5 show the student and instructor workflows respectively.

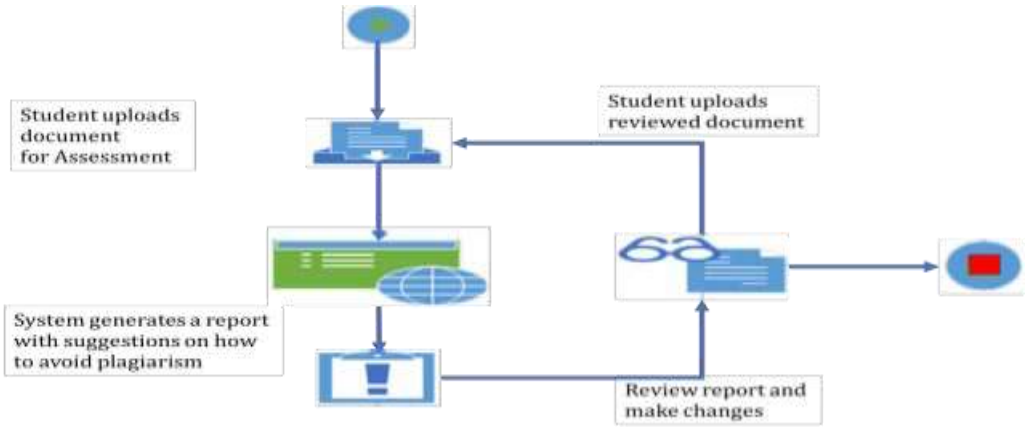

Figure 4: Student Workflow

The report generated by the system includes:

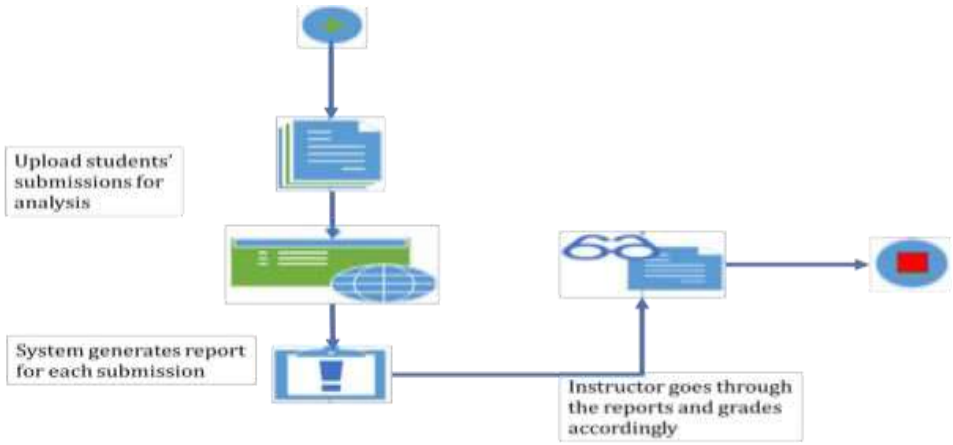

Figure 5: Instructor/Lecturer Workflow

- **Plagiarism index:** A score denoting how much of the document is a plagiarism.
- **Colour Coded Suspected areas:** Colours Ranging from light blue to Red. Denoting the degree to which the highlighted area is a plagiarism.
- **Helpful tips** on how to prevent plagiarism. Example explanations on how to properly reference an article (students only).
- **Hyperlinks** links to the plagiarized documents in each section (instructors only)

IMPLEMENTATION

a. Tools

The system was implemented based on the system flow in Figure 2. As discussed, there are several components, a pre-process that cleans up the document, a post-process that packages the result to an easily understandable form for the user, however, the major distinguishing functionality of this system is the NLP Model. This component is responsible for the generation of word vectors based on trained knowledge about how words are related to each other. These word vectors are then subsequently used to compare 2 documents to get their contextual similarity. The application was written as a web-based solution in Python 3 (Python, 2021) using PyCharm running on Anaconda. The model was also implemented in Python 3 using Spacy, an open source library used for advanced NLP, Pandas and NumPy. The development and testing of the application and model were performed on a system configured with Intel Core i7-7820HK CPU @ 4.2GHz, NVIDIA GeForce GTX 1070 GPU, 16GB of RAM and Windows 10 operating system.The testing application was written in Angular 7, HTML and CSS and implemented on JetBrains WebStorm 2019.1.4. This web page connects to a Python 3 backend application, hosted and deployed using Flask web framework. PyCharm for Anaconda provides useful debugging and data visualization tools, which were useful in testing our AI model. Finally the A Corpus of Plagiarized Short Answers (ACOPSA) created by Paul Clough and Mark Stevenson was used as dataset (Clough and Stevenson, 2011).

b. Data Analysis

Unlike in the case of a classification problem, a classifier has an inherently correct value that it can compare its generated value to. In such case, useful values such as accuracy, precision and recall can be calculated, but in this case, it would be impossible to calculate the “Accuracy” of a model, because the constituent of ”right” or ”wrong” are qualitative rather than quantitative.

A Corpus of Plagiarized Short Answers (ACOPSA) contains 100 documents, 95 answers provided by the 19 participants to 5 questions, and 5 documents that mark a source that may be plagiarized to a certain degree. The documents are labelled or categorized into 1 of 4 groups:

- *Non:* Participants here were providing with learning materials or textbooks and instructed to give a solution from any other sources, but not Wikipedia. (The original Texts where obtained from Wikipedia).
- *Light:* Participants were ask to base their answers in some way off the Wikipedia article, this means that they were allowed to manipulate the article by replacing texts or meaning.
- *Heavy:* Participants were once again asked to base their answers on the Wikipedia article, but could strictly only change their answers by rephrasing and keeping the same meaning, or merging 2 sentences in the original to one sentence in the answer or splitting a sentence in the original to multiple in the answer.
- *Cut:* Here participants were instructed to copy and paste parts of the article to create their solution, although there was no specification on what parts to copy or how they were mixed in the final result.

Table 2 shows a snippet of the ACOPSA dataset. The dataset contains 100 documents and one CSV file with information about the documents, the Name, Group Number, the Person who wrote the Paper and so on. The category shows which group the paper falls under, this will give us an idea of the expected output from a good plagiarism checker.

Table 2: Snippet of Input Dataset for ACOPSA

	File	Group	Person	Task	Category	Native Eng	Knowledge	Difficulty
2	g0pA_task	0	A	a	non	native	1	1
3	g0pA_task	0	A	b	cut	native	4	3
4	g0pA_task	0	A	c	light	native	5	3
5	g0pA_task	0	A	d	heavy	native	3	4
6	g0pA_task	0	A	e	non	native	4	3
7	g0pB_task	0	B	a	non	native	2	1
8	g0pB_task	0	B	b	non	native	3	3
9	g0pB_task	0	B	c	cut	native	5	3
10	g0pB_task	0	B	d	light	native	2	2
11	g0pB_task	0	B	e	heavy	native	4	3
12	g0pC_task	0	C	a	heavy	native	4	3
13	g0pC_task	0	C	b	non	native	3	3
14	g0pC_task	0	C	c	non	native	2	4
15	g0pC_task	0	C	d	cut	native	1	5
16	g0pC_task	0	C	e	light	native	2	2
17	g0pD_task	0	D	a	cut	non-native	1	1
18	g0pD_task	0	D	b	light	non-native	2	2
19	g0pD_task	0	D	c	heavy	non-native	3	3
20	g0pD_task	0	D	d	non	non-native	2	3
21	g0pD_task	0	D	e	non	non-native	3	3
22	g0pE_task	0	E	a	light	non-native	1	1
23	g0pE_task	0	E	b	heavy	non-native	2	2
24	g0pE_task	0	E	c	non	non-native	2	2

Extracting Useful Information

The only information relevant to this model, would be the file name FILE, and the question number TASK, i.e. the document to be compared against. All questions numbered 'a' will be compared against the first Wikipedia article, b with the second and so on. A snippet of this is shown in Table 3.

Table 3: Snippet of dataset after feature selection

16	g0pD_taskb.txt	b
17	g0pD_taskc.txt	c
18	g0pD_taskd.txt	d
19	g0pD_taske.txt	e
20	g0pE_taska.txt	a
21	g0pE_taskb.txt	b
22	g0pE_taskc.txt	c
23	g0pE_taskd.txt	d
24	g0pE_taske.txt	e
25	g1pA_taska.txt	a
26	g1pA_taskb.txt	b
27	g1pA_taskc.txt	c
28	g1pA_taskd.txt	d
29	g1pA_taske.txt	e
30	g1pB_taska.txt	a
31	g1pB_taskb.txt	b
32	g1pB_taskc.txt	c
33	g1pB_taskd.txt	d
34	g1pB_taske.txt	e
35	g1pD_taska.txt	a
36	g1pD_taskb.txt	b
37	g1pD_taskc.txt	c
38	g1pD_taskd.txt	d
39	g1pD_taske.txt	e
40	g2pA_taska.txt	a
41	g2pA_taskb.txt	b
42	g2pA_taskc.txt	c
43	g2pA_taskd.txt	d
44	g2pA_taske.txt	e
45	g2pB_taska.txt	a
46	g2pB_taskb.txt	b
47	g2pB_taskc.txt	c
48	g2pB_taskd.txt	d
49	g2pB_taske.txt	e
50	g2pC_taska.txt	a
51	g2pC_taskb.txt	b

OBTAINED RESULTS

After passing the test data through our model , each document was assigned a similarity score, which depicts how much the document is a plagiarism of the original content. The Similarity index of a document is the percentage of text in that document that was found to exist in other documents. This is no automatic indicator of plagiarism as the similar texts could have been quoted or properly cited. However, a very high similarity index is usually a sign of plagiarism in the document. Similarity between 2 documents is calculated as the similarity index. This does not tell us anything further than "Document A and B have X% of words in common". Similarity index is what existing plagiarism checkers generate, it is then down to the instructors to decide whether the author cited his sources properly, and if the sources referenced can legally be used. Using AI, this decision could be made by the computer which after analysing the texts, the citations and references, generates a "plagiarism index". This is a score denoting how much content in the document is plagiarized. Most plagiarism detectors are yet to implement NLP technologies, with only a few scratching the surface. This is the distinguishing factor of our system.

Table 4: Snippet of obtained result from our model

9	non	24.345849034080686
10	non	25.302392348096568
11	cut	77.18471181415273
12	cut	80.9225971659853
13	heavy	47.39293584553963
14	non	8.140950712206196
15	non	15.72068292759825
16	heavy	52.92181746611087
17	non	2.8651049513301174
18	cut	97.04827123872896
19	non	20.54539896852143
20	heavy	42.597662010164726
21	light	52.36175037371281
22	non	7.375546398835678
23	non	11.324280388454119
24	non	18.301902060852758
25	heavy	53.044671670231914
26	light	20.51011663343763
27	cut	62.91712808630842
28	light	43.522320873473
29	cut	45.96281117435829
30	non	33.203082599898906
31	non	11.25825038757575
32	heavy	25.43555612449901
33	non	18.766608749394667
34	heavy	48.40459980817878
35	light	66.03287877173405
36	non	12.609938402405307
37	non	13.746683758907038
38	non	18.135833450090878
39	cut	80.77759536125659
40	cut	65.03735910920054
41	light	27.598658704579982
42	heavy	61.532730212593876
43	non	23.794202008459
44	cut	77.03839619398224

This similarity score was then mapped to the Category the data came from. For instance, in Table 4, the document at index 22 had a plagiarism score of 7.37%, hence had no direct plagiarism from the original content, while indexed 13, was heavily plagiarism with a score of 47.39%, index 11 and 12 were direct cut with scores of 77.18% and 80.92% respectively.

In terms of evaluating the performance of the model, a method of converting the categories (heavy, light, etc.) was adapted to numerical values. These values would be spread over the possible result space ranging from 0-100%. Hence, Non would be assigned a value of 0, Light would be assigned 33% or 0.33, Heavy would be assigned 66% or 0.66, and Cut would be assigned 100% or 1. These values allow us to properly conceptualize our data, to see just how high or low the values should be for any given category. This means that in this case, a document labelled 'Heavy' with a value of 0.6 is probably bad, but just how bad?

To answer this, the model was considered to be a form of classifier. Hence, calculating the extent is simply a matter of getting the ration of all documents that were categorized correctly to all the documents in the corpus. The next question might then be, how do we determine if a document was correctly classified or not? It is important to note that these category values are arbitrary and were chosen to simply give a better view of how the values are distributed. The category value was then taken, and an arbitrary range around it was picked such that there is no collusion with other categories. A document is declared to be correctly classified if it falls within this arbitrary range. For example, one could say that all 'heavy' documents are correctly classified if the model returns any value between 0.5 and 0.7.

Model Validation

For better visualization, the results were colour coded, in a colour spectrum ranging from Blue to Red. Blue depicted low similarity values and Red implied high similarity index. This

is as shown on Table 5. Here, similar colours on both the category values and the plagiarism scores are expected.
Since the category values and plagiarism scores are meant to have a similar distribution, their similarity was calculated and used that as a means of validating our model. To accomplish this, the category values (from the dataset) and the plagiarism scores were converted to a vector. Afterwards, the distance between the two vectors were calculated. The cosine similarity would give a more appropriate value for the perplexity of the model. By running our model on the chosen dataset, a score of 0.9237 or 92.37% was obtained.

Table 5: Result snippet with colour intensity

15	non	0.00000	15.72068
16	heavy	0.66000	52.92182
17	non	0.00000	2.86510
18	cut	1.00000	97.04827
19	non	0.00000	20.54540
20	heavy	0.66000	42.59766
21	light	0.33000	52.36175
22	non	0.00000	7.37555
23	non	0.00000	11.32428
24	non	0.00000	18.30190
25	heavy	0.66000	53.04467
26	light	0.33000	20.51012
27	cut	1.00000	62.91713
28	light	0.33000	43.52232
29	cut	1.00000	45.96281
30	non	0.00000	33.20308
31	non	0.00000	11.25825
32	heavy	0.66000	25.43556
33	non	0.00000	18.76661
34	heavy	0.66000	48.40460
35	light	0.33000	66.03288
36	non	0.00000	12.60994
37	non	0.00000	13.74668
38	non	0.00000	18.13583
39	cut	1.00000	80.77760
40	cut	1.00000	65.03736
41	light	0.33000	27.59866
42	heavy	0.66000	61.53273
43	non	0.00000	23.79420
44	cut	1.00000	77.03840
45	non	0.00000	23.97656
46	non	0.00000	20.41310
47	non	0.00000	29.77394
48	light	0.33000	42.18579
49	cut	1.00000	80.76893
50	cut	1.00000	90.20230

Low  High

CONCLUSION

With plagiarism on the rise and available tools and technologies facilitating it, there is an urgent need for plagiarism detectors to evolve. Traditional word-based vector matching has several shortcomings that plagiarizers and paraphrasing tools can exploit. For example, though the phrases: “President greets the press in Chicago” and “Obama Speaks in Illinois”, have no words in common, they have very similar meaning. A plagiarizer who copies the first phrase off a source material and paraphrases to the phrase B would successfully outwit most plagiarism detectors of the day. In this work, we have developed a model, which incorporates artificial intelligence through natural language processing (NLP) to enhance plagiarism detection. As an example, using an NLP our model would be able to recognize the relationship between “Obama” and “President” as well as Chicago being a city in the state of Illinois. By establishing these relationships, the model can detect that the two sentences are contextually similar, despite not having any words in common.

Clearly, as sentences become less semantically similar, though implying the same thing, their plagiarism similarity index drops. The proposed model was validated by measuring the vector distance between the output of our

model (i.e., the plagiarism score) and the category value in the chose dataset. Obtain result using cosine similarity showed an accuracy score of approximately 92.4%. Based on this accuracy score, one can clearly see the impact of NLP on plagiarism detection.

Though this work has proposed and implemented a plagiarism checker powered by AI, there are some notable limitations, prominent among which was the small corpus size. Larger data sets would be required to test the scalability of the model and there are intentions to explore this in future works. Furthermore, cross lingual plagiarism check is also an avenue to explore in the future, as by infusing NLP, our model is prime for multi-lingual support. If the model is exposed to data in multiple languages, it should be able to understand the relationship between words across various languages.

REFERENCES

Ali, A.M., Abdulla, H., and Snášel, V. (2011). Overview and Comparison of Plagiarism Detection Tools. *DATESO. 161–172.*

Blackboard. Blackboard safeassign: A plagiarism prevention tool. https://www.blackboard.com/teaching-learning/learning-management/safe-assign. Last accessed: 26 March 2021.

Britannica. Definition of plagiarism. https://www.britannica.com/topic/plagiarism. Last accessed: 20 March 2021.

Checkforplagiarism. Checkforplagiarism.net. www.checkforplagiarism.net. Last accessed: 26 March 2021.

Clough, P. and Stevenson, M. (2011). Developing a corpus of plagiarised short answers. *Language Resources and Evaluation, 45(1):5–24*.

DeLong, D. (2012). Unintentional plagiarism. *Global Journal of Engineering Education, 4(1):137–155.*

Dr Dataman. Looking into natural language processing (NLP). https://dataman- ai.medium.com/, 2018. Last accessed: 26 March 2021.

Dreher, H. and Williams, R. (2006). Assisted Query Formulation Using Normalised Word Vector and Dynamic Ontological Filtering. *FQAS. Lecture Notes in Artificial Intelligence, 282–294*.

Foltýnek, T., Dlabolová, D., Anohina-Naumeca, A. et al. (2020). Testing of support tools for plagiarism detection. *International Journal of Educational Technology in Higher Education, 17(1):46*. https://doi.org/10.1186/s41239-020-00192-4.

Heinrich, E. and Maurer, H. (2000). Active documents: Concept, implementation and applications. *Journal of Universal Computer Science, 6(12):1197–1202*.

Hoad, T. and Zobel, J. (2003). Methods for identifying versioned and plagiarised documents. *Journal of the American Society for Information Science and Technology, 54(3):203–215*.

ICAI. International center for academic integrity. http://www.academicintegrity.org. Last accessed: 22 March 2021.

Ison, D. (2017). Academic Misconduct and the Internet. Handbook of Research on Academic Misconduct in Higher Education.

KIT. Jplag – detecting software plagiarism. https://jplag.ipd.kit.edu. Last accessed: 26 March 2021.

Kumar, A. (2021). The role of AI in plagiarized text. Learning Hub. https://learn.g2.com/ai-for- plagiarism, 2020. Last accessed: 26 March 2021.

Marcos. (2019). A beginner's guide to ruby on rails mvc (model view controller) pat- tern. https://hackernoon.com/beginners-guide-to-ruby-on-rails-mvc-model-view-controller- pattern-4z19196a, 2019.

McCabe, D. L. (2005). Cheating among college and university students: A north american perspective. *International Journal for Educational Integrity, 1(1):1-11.*

Meyer zu Eissen, S. and Stein, B. (2006). "Intrinsic Plagiarism Detection", in Proceedings of the 28th European Conference on IR Research (ECIR), Lecture Notes in Computer Science (LNCS), 3936: 565–569, doi: 10.1007/11735106_66.

Niezgoda, S. and Way, T. (2006). Snitch: A software tool for detecting cut and paste plagiarism. *ACM SIGCSE Bulletin, 38(1):51–55*.

Oladeji, F., Ajayi, O., Koleoso, R., Uwadia, C. (2018). Third eye – a plagiarism checker for academic theses. National Conference on Digital Inclusion: Opportunities, Challenges and Strategies, 27(1):225–236.

Pennycook, A. (1996). Borrowing others' words: Text, ownership, memory and plagiarism. *TESOL Quarterly, 30(2):201–230*.

Pertile, S., Moreira, V. P., & Rosso, P. (2016). Comparing and combining content- and citation-based approaches for plagiarism detection. *Journal of the Association for Information Science and Technology, 67(10), 2511–2526*. https://doi.org/10.1002/asi.23593.

PlagAware. Plagaware. www.plagaware.com. Last accessed: 26 March 2021.

Python. Python 3.0 release. https://www.python.org/download/releases/3.0, 2008. Last accessed: 14 March 2021.

Rezaeian, N. and Novikova, G. (2017). Detecting near-duplicates in Russian documents through using fingerprint algorithm simhash. *Procedia Computer Science, 103(1):421–425*.

RHIG (2021). Random House Compact Unabridged Dictionary. Random House Information Group. Last accessed: 26 July 2021

Shivakumar, N. and Garcia-Molina, H. (1999). Finding near-replicas of documents on the web. *Lecture Notes in Computer Science, 1590(1):204–212*.

Sulaiman, R. (2018). Types and Factors Causing Plagiarism in Papers of English Education Students. *Journal of English Education, 3(1):17–22*.

Szuchman, L. (2010) Writing with Style: APA Style Made Easy. Cengage Learning.

Turnitin. Turnitin. www.turnitin.com. Last accessed: 26 March 2021.

Vani, K. and Gupta, D. (2016). Study on extrinsic text plagiarism detection techniques and tools. *Journal of Engineering Science and Technology, 9(4): 2511-2526*.