International Journal of Advanced Statistics and Computer Science – Volume 2 Issue 2, May 2021

Available at www.ijascs.com

RESEARCH ARTICLE OPEN ACCESS

College Project Plagiarism Using Machine Learning Technique-Long Short-Term Memory and Recurrent Neural Network

[1]Vanzel D'silva, [2]Jubinu Varghese, [3]Shubham Yadhav, [4]Dr. Vaishali Jadhav

[1](Information Technology, St. Francis Institute Of Technology, Mumbai
Email: vanzeldsilva@gmail.com),
[2](Information Technology, St. Francis Institute Of Technology, Mumbai
Email: jubinuvarghese@gmail.com),
[3](Information Technology, St. Francis Institute Of Technology, Mumbai
Email: shubhamyada26@gmail.com),
[4](Information Technology, St. Francis Institute Of Technology, Mumbai
Email: vaishalijadhav@sfit.ac.in)

----------##########----------

Abstract:

Plagiarism is the act of stealing someone else's ideas or writings. Over the course of years, various plagiarism Software's have claimed that they are 100 percent accurate. However, these software's do nothing but use a series of hand-picked thresholds of metrics that ultimately determine the basis of whether or not we can claim that the said author is guilty of committing plagiarism. Since these thresholds are manually determined, they are not good enough to be able to detect all forms of plagiarism, which include unique situations. In this project, we will be making use of Recurrent Neural Networks(RNN) and the LSTM algorithm whose specialty is that it not only processes single data points (such as images), but also entire sequences of data (such as speech or video). Therefore, we will be detecting plagiarism using a pre-classified data set, and make use of deep learning to enable authorities without any knowledge of the methodology to detect plagiarism. **—LSTM, Deep learning, plagiarism, RNN**

----------##########----------

I. INTRODUCTION

The word "PLAGIARISM" refers to a piece of writing that has been copied from someone else and is presented as being your own work [2]. Today with the huge popularity of internet, so many documents are freely accessible. Now internet is an extensive source to collect data. Students can easily get their required information or data from internet. Plagiarism cases are an everyday topic, for example, in academics. So far, textual plagiarism has been the most common form of plagiarism. In this project, we will be working on textual as well as code plagiarism. Detecting plagiarism is an active field of research with a wide variety of available tools.

Detecting textual plagiarism is fairly easy, especially if the text has been copied verbatim. However, nobody copies texts verbatim anymore. But some smart student rearranged text from an active voice to a passive voice, or the other way around; it becomes difficult for the automated systems to detect the plagiarized material. A way to get rid of this issue would be to detect the meaning of the plagiarised text and compare it with the texts found on the web. However, detecting the semantics of the text is challenging for an automated system despite making use of structural and lexical similarities. Over the course of the years, many different approaches have been attempted to overcome these unique situations and challenges, but

only a little progress has been made so far as to achieve high levels of accuracy in the complex plagiarism methods.

Another factor that plays a role in plagiarism detection is the level of academic achievement. For example, a 3rd grader would have a comparatively lenient plagiarism detection process compared to a graduate student. Therefore, different levels in the academic system require different amounts of your own work to be created to certify as non-plagiarized and legit. The situation is very subjective and the amount of education and the preference of the respective professor is unique in every situation. Due to reasons like this - having a common, non-complex plagiarism detection system can be and effort proved to be futile. Due to reasons like this, it is a complicated and exhausting task to find a plagiarism software that adheres to one's particular requirements.

To combat this, now we introduce a system which will provides an easy approach to keep a track of plagiarism in document of student by taking input like code and proposal/report. Now before college select a project all the draft for the projects will be sent to a specific college UI and from there, they will be able to run the algorithm and get the similarity score between the given and past documents using ML and also tell whether to allow or not to allow that project. Machine learning is used to help users find settings that fulfil their requirements with an automatic procedure that is solely based on statistics from a data set made up of pre-classified plagiarized and non-plagiarized cases.

II. RELATED WORK

We are all well aware that there are way too many plagiarisms detection software's available out there. The ones that make use of simple computer algorithms, or to be more specific: cosine similarity. Cosine similarity is very popular amongst the typical plagiarism software's. While they are fruitful, there's one too many loopholes that allow a plagiarist to get away without being detected. Cosine similarity helps determine how similar two texts/documents are. The idea behind this is very simple. Let's take the example of two vectors that each represent a sentence. If the vectors are parallel in nature, or even close to being parallel, we can safely assume that the sentences are similar. Similarly, when the vectors are perpendicular to each other, this means the sentences are independent or dissimilar. However, cosine similarity is not good enough to cover all the unique situations that are put forth by plagiarists, which is the motivation for this research paper.

In 2015, Chitra and Anupriya Rajkumar published a paper "Plagiarism Detection Using Machine Learning-Based Paraphrase Recognizer"[1] where they used machine learning-based paraphrase recognizer, which operates by extracting lexical, syntactic, and semantic features, has been used to detect plagiarism in text passages.This Provides a good performance on various corpora, as well as the different categories of the P4P corpus, indicates the suitability of the current approach for detecting paraphrased plagiarism.

In 2016, Muna AlSallal, Rahat Iqbal, Saad Amin, Anne James and Vasile Palade published a paper " An Integrated Machine Learning Approach for Extrinsic Plagiarism Detection"[2] where they used Text representation for Extrinsic Plagiarism Detection.The experiments, the MCW usage patterns were captured to enhance class characterisation. The results revealed that LSA with adjusted weighting for stylometric features (MCW) performed better than traditional LSA.

In 2017, Shubhesh Amidwar published a paper " Plagiarism Detection Using Supervised Machine Learning Algorithm" .The proposed system makes use of supervised machine learning algorithm in contrast to the old algorithmic approach which is used in the existing system viz. Naïve Bayes is used.It displays the plagiarised text along with the author's name. Results display a text is plagiarized along with the name of the author whose text is copied and also the probability is displayed.

In 2019, El Mostafa HAMBI and Faouzia Benabbou published a paper " A Multi-Level Plagiarism Detection System Based on Deep Learning Algorithms".LSTM and CNN algorithms are used to detect Multi-Level Plagiarism Detection System.Tests on PAN Dataset show that our system is able to detect different types of plagiarism and specially the semantic one. NLP techniques to process and analyse the structure of documents.

III. RESEARCH METHOD

A. Authentication module

In authentication module, it consists of user, teachers and super admins it collects information like ID and password and compares it with the entries stored in database. If the data provides matches with the database, then the user is validated and directed to the specified UI. If he data provided does not meet the criteria then the validation is denied.

B. Login Module

Login module is a portal that allows users to login using their ID's and passwords. Here the student, teacher and super admins will login into the system through this login module. after you login you will be directed to the UI where you can upload the reports/proposals and also check if it's accepted or not and percentage of plagiarism.

C. Data Extraction

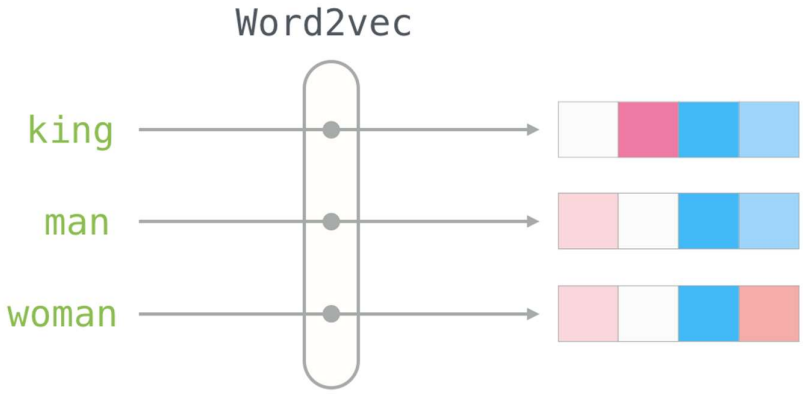
After uploading the file i.e. word or pdf extraction of text data using Optical Character Recognition takes place where it recognizes text within a digital image and it is commonly used to recognize text in scanned documents and images. The extracted data will be made available in formats like XML

D. Data cleaning

After data extraction next phase is data cleaning. Data cleaning is done to identify and remove stalk board, errors, duplicate words, incomplete data. This improves the quality of the training data and will help you to yield better results from the machine learning functions. Also, we streamline our data.

E. Vectorization

After data cleaning the data is converted into vector form. In vectorization, the letters are converted into number format. Word2Vec is used is to group the vectors of similar words together in vectorspace. That is, it detects similarities mathematically. Word2Vec creates vectors that are distributed numerical representations of word features, features such as the context of individual words which can help in detection of plagiarism.



F. ML Model

First we get data from the dataset then we perform data cleaning, data processing and convert it into vector format so that it can be easily read by the computer this all is done before training. Now in training we use Machine Learning algorithm like Long Short Term Memory and Recurrent Neural Networks. Here RNN uses sequential memory to memorize the previous information and make predictions while keeping that in mind. Let's look at an example, where we build a chatbot using a traditional neural network to find the intent of the user. We feed it with the following question: "What time is it?" Since RNN works sequentially, it gets access to one word at a time.

Given below are the steps for the same:

1. The word "What" is fed into the RNN. The RNN then encodes this word and produces an output.
2. Next, the word "time" is fed into the RNN but in a hidden state from the previous step. (The hidden state will represent information from the previous steps)
3. The word "is" is fed into the RNN.
4. The word "it" is fed into the RNN.
5. The "?" is fed into the RNN.
6. In the final step, the RNN collects information from all the previous steps, stores it in the last step as the final output, feeds it to the feed-forward layer and decides the intent based on all this information.

An RNN is trained using a back propagation algorithm. The back propagation is what causes the RNN to have short term memory and vanishing gradient. The short term memory and vanishing gradient causes the RNN to perform poorly. The vanishing gradient causes the RNN to not learn the long-range dependencies across the various time steps. Therefore, as it progresses, the word "what" and "time" are not considered when trying to predict the user's intention. The only thing left with the network is "is it?" to make a decision - which is pretty difficult even for a human being, let alone a neural network. Therefore, the RNN by itself proves to be incompetent in providing an answer.

This can be solved by utilizing a specialized recurrent neural network, known as Long Short Term Memory or LSTM. As the name suggests, Long Short term Memory is a specialized neural network that is used to combat the problem of Short term memory faced by a typical Recurrent Neural Network (RNN). An LSTM will essentially perform just like an RNN, except they are able to mitigate the problem of long term dependency using the "gate" mechanism. These gates know what information to add or delete to the hidden state and act as different tensor operations. This mitigates the problem of short term memory. By simple logic, it is evident that prediction can be made easier if there is sequential logic to the sentence. In a typical prediction problem, where we have a bunch of characters, the goal is to predict the next character of the sequence. However, this prediction is impossible to make without knowing the context.

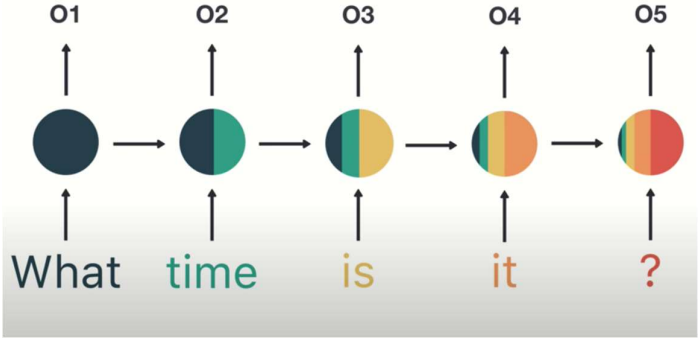


Fig 1: Steps of performing RNN on a simple sentence.

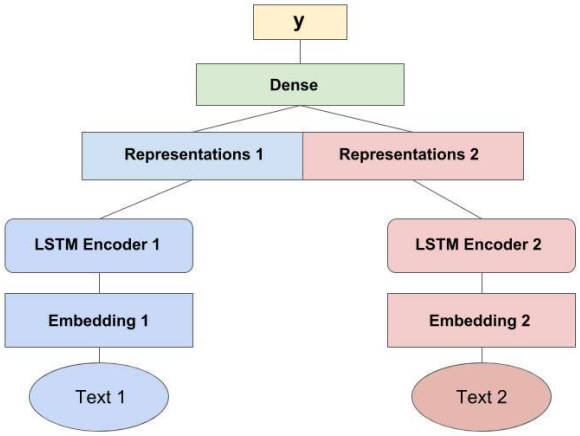


Fig 2: Step for calculating plagiarism

G. Data preprocessing

Then the load trained machine learning model passes the preprocessed i.e. the raw data is transformed to a useful and efficient format data to the model to get the output which is the percentage of plagiarism as it tells the proposal/report is accepted or not by the score set. Then the data is stored in the database and the output is sent to the student which can be viewed on the UI by both teachers as well as student. Teachers can also view the result and change the result.

IV. RESULTS

In order to determine the accuracy and suitability of our proposed approach for the plagiarism detection system, we made use of four different split corpora from Kaggle.com. Upon analysing one of the corpus, we found that there are around 7859 pairs of strange or suspicious passages, out of which 3792 passages are not plagiarized and the rest 4067 are plagiarized. The construction of the corpus was done after performing crowd sourcing on Amazon's Mechanical Turk. In this situation, various texts were extracted from a project and these texts were paraphrased. These texts passed the green check on the paraphrasing since they were reviewed to be very similar to the original by the system and were hence rejected. On the other hand, cases where the content was grammatically correct and had the same meaning as the source was accepted as a paraphrase.

The second corpus consisted of 1716 text articles that were extracted from an esteemed Press Association. Since the text articles are a rewritten version of the corresponding PA source: we had to categorize the texts into various categories that described whether or not they were fully derived. For this project, we chose 253 articles with a single source, and were categorized into "Plagiarized" and "Non-Plagiarized".

The third corpus consisted of 95 short answers that were gathered from participants who answered five different questions. These answers were originally labelled into 4 different categories but were then converted into the "Plagiarized" and "Non-Plagiarized" categories.

The measures considered to evaluate the level of plagiarism are based on Accuracy, Recall, and F-measure. Accuracy, Recall, and F-measure. They are calculated in the equations given below:

$$\text{Accuracy (A)} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision (P)} = \frac{TP}{}$$

$$\text{TP} + \text{FP}$$

$$\text{Recall (R)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F - measure (F)} = \frac{2 * \text{Precision} * \text{Recall}}{(\text{Precision} + \text{Recall})}$$

```
def initPlag(inputQuery,file_path):
    if True:
        universalSetOfUniqueWords = []
        matchPercentage = 0
        inputQuery = request.form['query']
        lowercaseQuery = inputQuery.lower()
        queryWordList = re.sub("[^\w]", " ",lowercaseQuery).split()
        for word in queryWordList:
            if word not in universalSetOfUniqueWords:
                universalSetOfUniqueWords.append(word)
        fd = open(file_path, "r")
        database1 = fd.read().lower()
        databaseWordList = re.sub("[^\w]", " ",database1).split()
        for word in databaseWordList:
            if word not in universalSetOfUniqueWords:




    queryTF = []
    databaseTF = []
        for word in universalSetOfUniqueWords:
                queryTfCounter = 0
                databaseTfCounter = 0
                for word2 in queryWordList:
                        if word == word2:
                                queryTfCounter += 1
                queryTF.append(queryTfCounter)
                for word2 in databaseWordList:
                        if word == word2:
                                databaseTfCounter += 1
                databaseTF.append(databaseTfCounter)
        dotProduct = 0
        for i in range (len(queryTF)):
                dotProduct += queryTF[i]*databaseTF[i]
        queryVectorMagnitude = 0
        for i in range (len(queryTF)):
                queryVectorMagnitude += queryTF[i]**2
        queryVectorMagnitude = math.sqrt(queryVectorMagnitude)
        databaseVectorMagnitude = 0
        for i in range (len(databaseTF)):
                databaseVectorMagnitude += databaseTF[i]**2
        databaseVectorMagnitude = math.sqrt(databaseVectorMagnitude)
        try:
            data = databaseVectorMagnitude
            classifier = joblib.load('app/lstm.pkl')
            queryVectorMagnitude = classifier.predict([data])
        except:
            pass
        matchPercentage = (float)(dotProduct / (queryVectorMagnitude * databaseVectorMagnitude))*100
        return matchPercentage
```

Fig 3: Page showing file content, code level

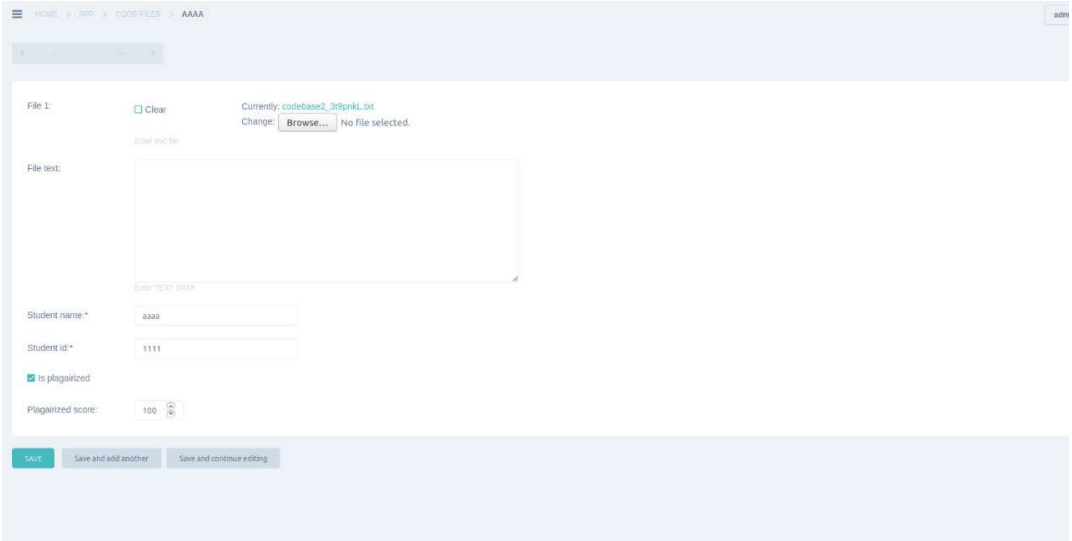


Fig 4: Page showing Input file from student

Fig 5: Plagiarism output, with green check indicating plagiarized content

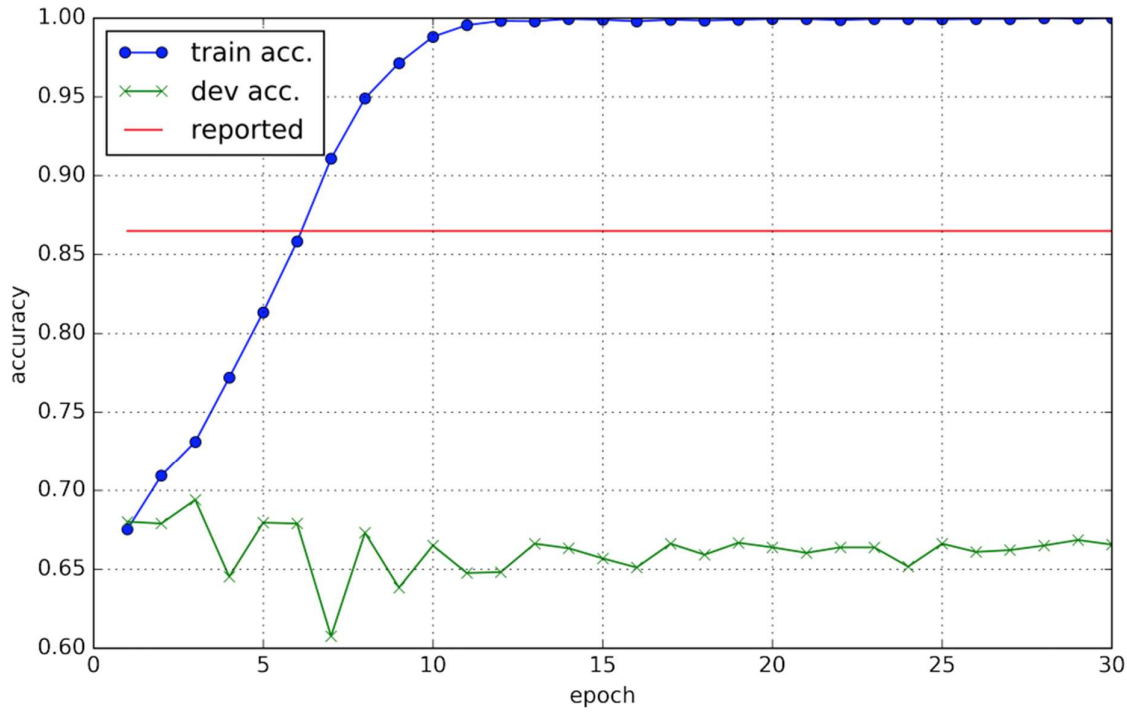


Fig 6: Train vs Dev accuracy

When previous models were trained on the MSRP corpus from kaggle.com: the classification results achieved were not random. Comparing with that scenario, we achieved similar results. Upon observing the figure, we can observe that the features learned during training are irrelevant to the dataset of dev. One might assume that it could be due to various features like a particular combination of words in the training set which are not present in the dev set like some topics, specific words, etc.

V. CONCLUSION

The system would prove to be an efficient system to obtain Plagiarism results in college projects. In today's world, plagiarism is a day to day occurrence. With how common it is, there are more

and more methods being invented to avoid getting caught by plagiarism tools. One of the ways of doing so is to summarize and paraphrase the content. Due to reasons like this, there is more need for a more suitable plagiarism detection system. It is necessary for this plagiarism detection system to make use of effective mechanisms for the automatic detection of plagiarism. In this project, the approach of paraphrase recognition is used. This is done so as to detect plagiarism in code as well as documented sources. We have utilized Recurrent Neural network using LSTM to combat this problem. The system has been tested used three different scenarios from the website: kaggle.com. The system is also performing better when compared with the best performing system on the previously mentioned corporas. Testing with various subcategories of P4P also gave out excellent results. This leads us to conclude that the paraphrasing recognition techniques shows a lot of potential with regards to plagiarism detection.

VI. FUTURE SCOPE

One of the directions of future work is we can make this application available for internet i.e a web application of plagiarism can be made. We will also work so that we can upload any file pdf word etc.

REFERENCE

[1] Chitra; Anupriya Rajkumar;, "Plagiarism Detection Using Machine Learning-Based Paraphrase Recognizer" J. Intell. Syst. 2015.

[2] Mausumi Sahu;, " Plagiarism Detection Using Artificial Intelligence Technique In Multiple Files" International Journal Of Scientific & Technology Research vol 5, issue 04, April 2016.

[3] Upul Bandara; Gamini Wijayrathna;, "Detection of Source Code Plagiarism Using Machine Learning Approach" International Journal of Computer Theory and Engineering, Vol. 4, No. 5, October 2012.

[4] Francisco Rosales; Antonio García; Santiago Rodríguez; José L. Pedraza; Rafael Méndez; Manuel M. Nieto;" Detection of Plagiarism in Programming Assignments", IEEE Transaction On Education, vol. 51, no. 2, May 2008.

[5] Shubhesh Amidwar;, " Plagiarism Detection Using Supervised Machine Learning Algorithm" JETIR, Volume 4, Issue 06 June 2017.

[6] Ethan Hunt; Ritvik Janamsetty; Chanana Kinares; Chanel Koh; Alexis Sanchez; Felix Zhan; Murat Ozdemir; Shabnam Waseem; Osman Yolcu; Binay Dahal; Justin Zhan; Laxmi Gewali; Paul Oh;," Machine Learning Models for Paraphrase Identification and its Applications on Plagiarism Detection", IEEE International Conference on Big Knowledge, 2019

[7] El Mostafa HAMBI; Faouzia Benabbou;, " A Multi-Level Plagiarism Detection System Based on Deep Learning Algorithms", IJCSNS International Journal of Computer Science and Network Security, VOL.19 No.10, October 2019

[8] S. Priya; Anukul Dixit; Krishanu Das; Ronak Harish Patil;, "Plagiarism Detection in Source Code Using Machine Learning", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249-8958, Volume-8 Issue-4, April 2019

[9] Mumthaz Beegum.M; Aji S;," A Method for Text Plagiarism Classification Using Deep Learning", JASC, November 2017

[10] Muna AlSallal; Rahat Iqbal; Saad Amin; Anne James; Vasile Palade;, " An Integrated Machine Learning Approach for Extrinsic Plagiarism Detection", 9th International Conference on Developments in eSystems Engineering, 2016.

