EDUEASY – SMART LEARNING ASSISTANT SYSTEM

Project ID: 2020-009

Project Final Report (Draft)

IT117106634 – Nuwanjaya L.A.P.Y.P

B.Sc. (Hons) Degree in Information Technology

Department of Software Engineering

Sri Lanka Institute if Information Technology

Sri Lanka

September 2020

DECLARATION

I declare that this is my own work and this report does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name	Student ID	Signature
Nuwanjaya L.A.P.Y.P	IT17106634	

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

Signature of the supervisor Date

....

(Dr. Anuradha Karunasena)

ABSTRACT

Students who study smart instead of study hard, can have good success in their education. But studying smartly is a challenge without a correct support. University students get used to study their lessons in various ways. Students who do not go lectures regularly may not have good lecture notes. Those who don't have a proper lecture note, may feel uncomfortable near an exam. EduEasy is an E-learning application for students to effectively learn and revise lectures done at university. EduEasy application generates a summarized note by converting lecturer's voice into a text document which is recorded at the lecture time. Students can directly navigate to the lecture note, references and sample questions relevant to the lesson. Slide Matcher component navigates to the exact place of lecture note from the summarized note according to the selected part of summarized note. It may help students to save the time which is wasted to find lecture notes by manually. If students want a more explanation about some parts of lecture which can't be understood by only referring the summarized note, can use this Slide Matcher tool to find the lecture note easily. The Slide matcher tool is basically done with text similarity and text classification. Both Lexical similarity and Semantic similarity should have captured when calculating the similarity. Natural Language Processing functions were used since this research is basically done with human languages. TF-IDF algorithms (Vector Counter Pairwise) and Gensim were used to calculate the similarity of words.

This is mainly focused on Text similarity and text mining technologies which calculate the similarity of texts.

Keywords— Vector Counter Pairwise; Gensim; Lexical similarity; Semantic similarity; TF-IDF algorithms; Natural Language Processing

ACKNOWLEDGEMENT

Working on this project on “Edu-Easy E-learning system” was a source of immense knowledge to me. I would like to express my thanks of gratitude to all those who provided me the possibility to complete this project. A special gratitude I give to Dr. Janaka Wijeykoon, the Lecture-in-charge of Comprehensive Design and Analysis Project and Dr. Anuradha Karunasena, our project supervisor and Ms Pradeepa Bandara, our Co-Supervisor whose contribution in stimulating suggestions and encouragement helped me to complete the research project.

The guidance and support received from my family, friend and my group members was more helpful to finish the project successfully. Furthermore I would like to acknowledge all the others who offered their fully support on completion of this research project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

TF-IDF	Term Frequency – Inverse Document Frequency
NLP	Natural Language Processing

1. Introduction

1.1 Background

A person has been learning since his birth in many ways by passing various steps. Baby's mother is the first teacher and then he or she enters to the school as a school child. Teachers who are working in school guide that child how to identify objects, how to write letters in correct way and so on. Child cannot learn an algorithm or a formula by his own during this time period. A teacher's guidance is essential to learn as a primary school child. When this child is growing and after turning into 12 or 13 years, trying to learn new subjects, languages, technologies by own. During the school time period of a child, teachers or his parents are always being with child guide his to the correct path. Students must attend to all classes at school. Teachers identify the weak students, conduct extra classes for them and so on.

But when it comes to the university environment it is completely difference from the school. Lecturers only conduct the lectures. Students should collect the additional knowledge by referring reference books, searching on internet and etc. Normally lectures don't give a note during the lecture time. Students should make notes by their own. If a student couldn't attend to the lecture, he would not be able to learn that lesson. Most of the students are used to study for exams when the exams are near. They are trying to find notes, answers and lecture notes. Students waste their time to find those notes and answers. Developing a web application which can be more helpful to university students to complete their academic activities and overcome above mentioned problems is very essential.

EduEasy is an E-learning application for students to effectively learn and revise lectures done at university. It has mainly four components. They are Note Taker, Reference Finder, Question Finder and Slide Matcher. When the lecturer is doing the lecture, the voice of lecturer is recorded by a microphone and it will convert into a text document as a lecture note. Then unwanted information will be removed and a note will be created called a summarized note. Students who missed the lectures can refer this summarized note and can study by himself.

Slide Matcher component matches the relevant lecture slide content according to the generated summarized note. It means If student wants to refer the lecture note which is published by the lecturer, can navigate to the lecture note directly using slide matcher tool. Student has to click on the topic or the sub topic of summarized note and system will display the exact place where the lecture content is.

Finding the exact lecture note according to the summarized note is basically about matching the words and find the similarity of documents.

Similarity

According to the Cambridge dictionary, Similarity is the fact that people or things look or are the same. Similarity is the state of being almost the same, or a particular way in which something is almost the same. Finding the text similarity between two documents can be done in many ways.

- Using string matching algorithms

String matching algorithms such as Rabin-Karp algorithm, Boyer – Moore String-search algorithms basically matches the strings of a word with each words of other sentence and find the similarity.

- Using Natural Language Processing (NLP) functions
  There are some functions in NLP which can be re-processed the data. Re-processing data is very essential to find the similarity of two text documents which are converted to text from voice. Basically Edu-Easy application works with lecturer's voice which is converted to text. Then the converted words can be changed to it's based form or root form and changed prefixes and suffixes. NLP functions such as tokenization, Stemming, Lemmatization will help to overcome those challenges.

- Using tf_idf Vectorization
  tf_idf is an abbreviation for Term Frequency – Inverse Document Frequency which is very common algorithm to transform text into numbers and identify the similarity. Tf-idf is a measure of originality of a word by comparing the number of times a word appears in a document with the number of documents the word appears in.

- Using Gensim
  Genism is a popular open source natural language processing library which can perform with complex texts.

This document provides a clear explanation about research problem, solution and objectives including literature survey and intelligible description of methodologies used for the implementation of EduEasy system. Machine Learning and Natural Language Process technologies are described with respect to the research problem. Research problem is described further based on important factors extracted from Research gap and literature survey

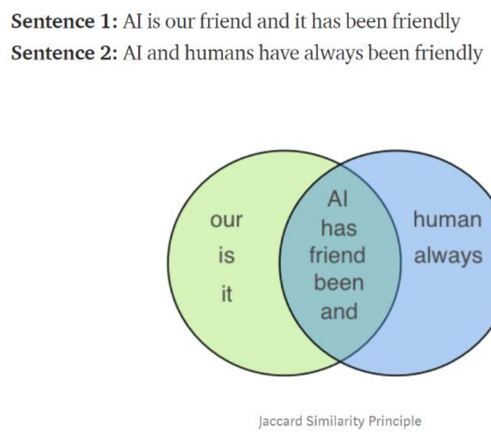
1.2 Literature Review

Matching the relevant lecture slide content according to the generated summarized note is the main objective of this Slide Matcher component. Simply system compares the words of topic which is in the summarized note with lecture content. By developing an algorithm to match the string values can solve the problem. But there can be many failures by only using string matching algorithms. In String matching algorithms, it takes the values into an array and compares with the other variables. But comparing exact words cannot find the best solution. Because EduEasy system converts the lecture's voice into text document, removes the unwanted phrases and summarizes the note. Therefore the topics and subtopics in summarized note might not be exactly same in lecture note.

Then a problem occurs how to compare each words without having same words. There can be the same meaning in different words.   By referring research papers, a solution was found called Natural Language Processing which was found by Richard Bundler and John Grinder. A research[1] done about Natural Language Processing algorithms described deeply about the functions of NLP and how it can be used to re-process the data. NLP is a part of computer science and AI which deals with human languages. Chatbox, Spell Checking , Keyword searching are some applications which used NLP algorithms.

There are many functions in NLP which helps to re-process the data. The research[1] has explained about the main steps and clear solutions to overcome the problems if only string matching algorithms are used to compare the similarity of two sentences. Data cleaning is a function about to clean the data. Get the data in clean and standard format for further analysis is very important. This function can remove the unwanted spaces, commas and etc. After getting the data cleaned, system cannot match the whole document at once. So there is an another function called Tokenization to reduce the complex of sentences. Tokenization can break a complex sentence into words, understand the importance of each word to the sentence and produce a structured description on an input sentence. Next function is Stemming. Stemming does normalize words into its base form or root form, cuts the prefixes and suffixes. Changing affected, affective to affect is an example. There is an another function called Lemmatization which groups together different infected forms of a word called Lemmas and outputs a proper word. Using Lemmatization the words 'go', 'going' and 'gone' will turn into 'go'. Chuncking is an another function which is picking up individual pieces of information and grouping them into bigger pieces such as nouns, verbs and etc. POS tags and Named Entity Recognition are some other functions in NLP which helps to reprocess the data.

An another research[2], published by Aselsan research center in Turkey described about the types of text similarity and how it helps to find the most similar sentences . It describes about mainly two types in text similarity. Lexical similarity and Semantic similarity are the main two types. Lexical similarity is about surface closeness and semantic similarity is about the meaning of the sentence. When we measure the similarity, both similarities should be noticed. An article[3] explained about this problem with an example. There are two sentences called "The cat ate the mouse" and "Mouse ate cat's food". If the word level similarity is only concerned, these two phrases appear very similar as 3 of the 4 unique words same. But when we consider the meaning, it is totally different. To overcome this problem there is an algorithm called Jaccard similarity which is explained in an article[3].



Sentence 1: AI is our friend and it has been friendly
Sentence 2: AI and humans have always been friendly

figure 1.1 – Jaccard similarity principle

This figure 1.1 is shown an explanation about how Jaccard similarity works.

There is an another problem may occur when comparing two sentences. There can be two sentences, both sentences have the same meaning but the words used in two sentences are exactly different. Then when we compare the similarity, both sentences do not match with each other. For exam a research paper[4] explained it with an example. “President greets the press in Chicago” and “Obama speaks in Illinois” are those two sentences and to identify the comparison of those two sentences that research paper[4] explains an algorithm called K-mean algorithms and Hierarchical Clustering Dendrogram. Here these two sentences should be converted into vectors first and there are methods explained “Count Vectorizer method” and “Word Embeddings”.

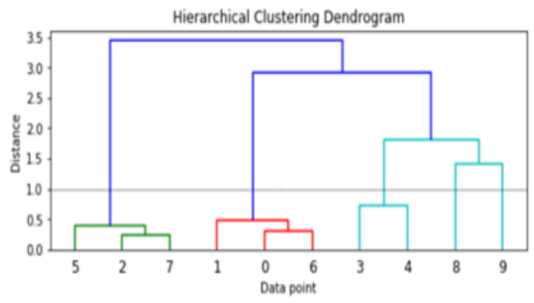
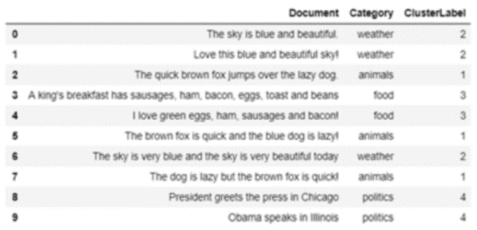


Figure 1.2 – Hierarchical clustering dendrogram



	Document	Category	ClusterLabel
0	The sky is blue and beautiful.	weather	2
1	Love this blue and beautiful sky!	weather	2
2	The quick brown fox jumps over the lazy dog.	animals	1
3	A king's breakfast has sausages, ham, bacon, eggs, toast and beans	food	3
4	I love green eggs, ham, sausages and bacon!	food	3
5	The brown fox is quick and the blue dog is lazy!	animals	1
6	The sky is very blue and the sky is very beautiful today	weather	2
7	The dog is lazy but the brown fox is quick!	animals	1
8	President greets the press in Chicago	politics	4
9	Obama speaks in Illinois	politics	4

Figure 1.3 – Data clustering example

This figure 1.2 is shown about Hierarchical Clustering Dendrogram a figure 1.3 is shown how clustered words which has the same meaning with difference words as “Obama” and “President” are grouped in politics category.

A research paper[3] describes an mathematical algorithm called Cosine vector similarity to calculate the semantic similarity. It is an algorithm which calculates the similarity of a sentence by measuring the cosine angle between two vectors.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

Figure 1.4 - cosine vector algorithm

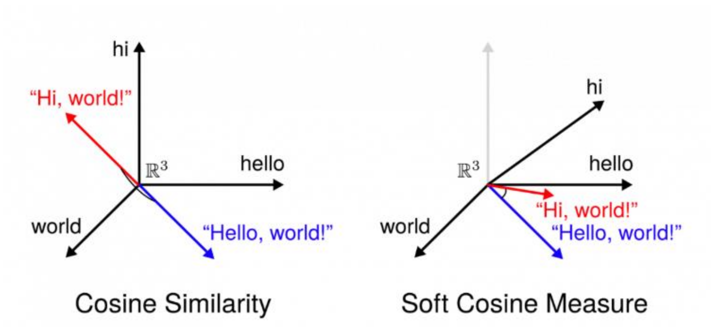


Figure 1.5 – example of cosine vector similarity

This figure 1.5 is explained about Cosine vector similarity and how it can be graphically drawn.

A research paper [5.1] which was written about text similarity in vector space model, has been described two ways how text similarity can be calculated. Structure based and structure agnostic are the two ways. Structure based means rely on a logical structure of the text and transform it into an intermediate structure. Then the comparison will be done. Second way is Structure based which ignores the structure. A vector Space Model(SVM) is used to represent the text. Then that text is converted into a numeric vector. Research paper[5] described about TFIDF vectorization model which measures the term frequency of each term in a text and multiplies it by the logged inverse document frequency of that term across the entire corpus.

After converting text to a vector, all texts are represented as low dimentiol vectors. Given two vectors can measure similarity in many ways. According to the paper [5] Euclidean distance, angular distance, correlation are some of them. Even though these difference methods measure the similarity in difference ways, research paper [5.1] adopted Cosine Vector similarity as the well-known and frequently used algorithm. Author discussed about how those algorithms represent to similarity detection power for texts. Doc2Vec variants and Word2Vec variants are compared and author found that Doc2Vec is more superior than Word2Vec. There are less researches conducted about the performances of similarity comparison for longer texts according to the author [5.1]. But Using TFIDF methods to detect the similarity is quite advanced because it has the advanced word embedding techniques.

There is an another research [6] has been conducted about measuring the text similarity. It is divided string based text similarity algorithms into two parts; Character based and Term based. N-gram, Jaro, LCS, Damerau-Levenshtein are some character based string algorithms and Block Distance, Cosine Similarity, Jaccard and Overlap Coefficiant aresome Term based String algorithms.

There are many applications and tools which helps to find the similarity of sentences and paragraphs. Some of those tools are used NLP algorithms while others are using String matching algorithms to compare. Here I compared the existing products with our proposed Slide Matcher tool.

	Copyleaks	TextAnalizer	CountWordsFree	Chatbox	Information extraction	Slide Matcher
Check the exact similarity of given sentence	✓	✓	✗	✗	✓	✗
Find similarity of paragraph using the frequent of given unique key word.	✗	✗	✓	✗	✗	✗
Re-process data using NLP algorithms	✗	✗	✗	✓	✓	✓
Use Jaccard similarity to find the correct words order of compared sentence	✗	✗	✗	✓	✓	✓
Use K-mean algorithms to analyze the meaning of words.	✗	✗	✗	✓	✓	✓

1.3 Research Gap

The Slide Matcher compares the selected topic or sub topic with the lecture notes which are stored in cloud database. Normally the system does that the words or letters which are in the selected topic are taken to an array and try to match with the most similar words from the database. There are some researches which are done about string matching algorithms to compare the similarity of two words and two sentences. if the system takes word by word in a sentence and stores those words in an array and try to find the similarity, it might take too much time and will produce wrong output because of Stemming and Lemmatization as discussed in literature survey [1].

Data which are going to be compared, should be in a structured form. Then it will easy to deriving meaningful information from natural language text. There are some tools, used string matching algorithms to compare the similarity like CopyLeaks and TextAnalizer. These tools try to compare the similarity without doing any re-processes. It just finds the similarity of given word using string matching algorithms. There is an another tool called CountWordsFree which shoes the count of unique word of a paragraph. We can't find the exact lecture note by counting the unique key words. Because sometimes there won't be that key word in the lecture note.

There are some tools and applications which use Natural Language Processing functions to compare the similarity. They are Keyword searching, spell checking, Sentimental analysis, Chatbox, speech recognition, Information extraction. Natural Language Process tries to understand the data by Planning the text, Planning the sentence and Realizing the text. Then it will produce the most correct output. Instead of using string matching algorithms, using NLP algorithms[1], Jaccard similarity[4], cosine vector similarity[3] we can compare the similarity of the sentences produce correct and efficient output.

But the summarized note of EduEasy application is generated using the lecturer's voice. So using a technology which can deal with human languages, is very important to calculate the similarity of this kind of documents. And choosing a technology, which can capture lexical similarity an semantic similarity, is also very important. Because the meaning of the word and the surface clossness can change the test result.

Slide matcher tool find the most similar lecture slide according to the summarized note. Normally students are finding the lecture notes manually. But EduEasy system proposed the Slide Matcher tool which can find the most similar lecture slide automatically.

2. RESEARCH PROBLEM

Students who study smart instead of study hard, can have good success in their education. But studying smartly is a challenge without a correct support. University students get used to study their lessons in various ways. Students who do not go lectures regularly may not have good lecture notes. Those who don't have a proper lecture note may feel uncomfortable near an exam. EduEasy is an E-learning application for students to effectively learn and revise lectures done at university. EduEasy application has mainly four components. They are note taker, reference finder, question finder and slide matcher.

Students who don't have proper notes, may feel unconfutable during the exam period because without having a good note it is very difficult to understand the theories and learn them. Then they waste their time to find out good notes. EduEasy application generates a summarized short note by recording the lecturer's voice and converting it into a text document. Then using some algorithms unwanted data will be removed and generates a summarized note. After approving the summarized note by the lecturer, it will be uploaded to the system and student can refer it.

While referring the summarized note students may want to refer some questions or regarding to the relevant part. Then students should have to find sample questions by manually searching in search engines. Then the students time will waste and sometimes they might not find sample questions. EduEasy application has a method to find the questions directly. Question finder component directly navigates to the websites which has sample questions regarding to the relevant lecture part.

Referring the relevant reference books is very important as undergraduates. But most of the students skip to refer the reference books. Lecturer can't teach everything in the lecture time. Students should search more information by themselves. Our proposed system has a tool to navigate to the relevant references directly from the summarized note, It will help students to find references easily.

Normally students try to find the lecture notes by manually when it comes near to an exam. Sometimes they don't have lecture notes with them or the system which is maintained by university might be removed lecture content from the system. But the Slide Matcher tool can find the correct lecture note navigate directly to the relevant content. It will save the time of students and students can study smartly by comparing summarized note and lecture note.

The proposed system will help to overcome those problems which are explained above. Students can study easily without wasting time to find short notes, sample questions, references, lecture notes.

3. OBJECTIVES

3.1 Main Objective

Matching the relevant lecture slide content according to the generated summarized note.

The summarized note is generated based on lecturer's voice which is recorded in the lecture time. When students are referring the summarized note, he or she wants to read the relevant lecture slide for better explanations or clarify the doubts. Then students normally try to find the lecture slides manually. Sometimes they couldn't find the lectures because of some reasons. Then students waste more time to find the lectures than studying. So reducing the wasting time of students and studying smart and comfortable is the main objective of Slide matcher tool.

3.2 Specific Objectives

- Upload the lecture PPTs to the system.
  All the lectures should be uploaded to the system. This can be done by the lecturer or any responsible person. All the lectures should be uploaded in pptx format. Then uploaded lectures should be stored in the cloud database. And it can be retrieved while the similarity algorithm is working.

- Capture the summarized note content
  When the user uses the slide matcher tool, he can be able to copy the summarized note content and paste it in the slide matcher tool for finding the relevant lecture. Sometimes user does not want to copy the whole content but the sub part only. Then the tool can be able to identify that function too.

- Implement an algorithm to calculate the similarity of given document
  Calculate the similarity of summarized note and other lecture ppts and find the most similar lecture ppt is one of the main objective of Slide Matcher tool. Implemented algorithm should be able to identify the human languages.

4. METHODOLOGY

This section describes how “Slide Matcher” will be designed and implemented. This is mainly considered of the process on measure the text similarity and the flow in the component.

4.1 System Overview Diagram

Slide Matcher component has mainly two algorithms which measure the text similarity. Necessary databases are defined at the beginning of the design process. Measuring the text similarity is implemented using two different algorithms based on NLP and text mining. Each algorithm is tested against software quality measurements and most productive way is included to the integrated system.

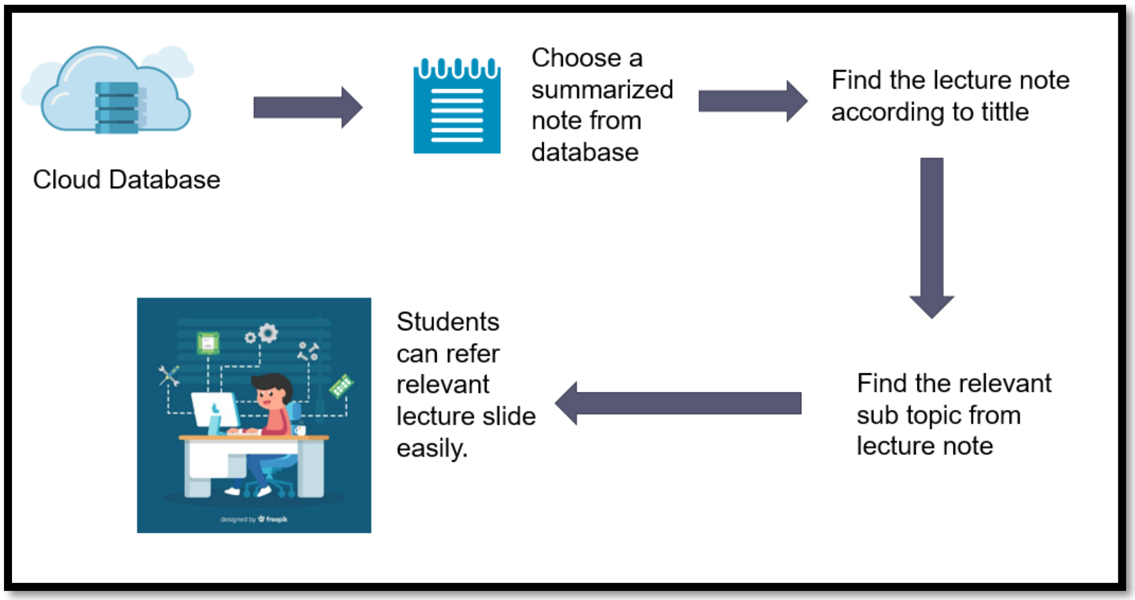

Cloud Database
Choose a summarized note from database
Find the lecture note according to tittle
Find the relevant sub topic from lecture note
Students can refer relevant lecture slide easily.

Figure 4.1- System overview diagram

4.2 Requirement Gathering and Analysis

Requirement gathering and analysis is the most crucial part in any software development project. Especially when it is come to a research, requirement gathering and analysis is the core, since solution for the problem isn’t known. Thus a requirement should be unambiguous, clear, feasible and correct. Proper knowledge about the research area is very important.

Before proposing an algorithm for measuring the similarity between summarized note and lecture contents, it was very important to understand the current technologies based on NLP, Text mining and topic modeling. Since the summarized note is generated based on lecturer’s voice, I need to have a good understand about the technologies which are developed to understand the natural language used by human beings. Also by studying Existing systems, Reading blog posts and

Literature reviews & Research Papers. A discussion with university students was conducted to identify the problems about finding lecture notes manually.

By further analysis of the gathered information, the features are clearly distinguished and the functionality of the component is carefully defined, once again taking into consideration the identified elements of the final outcome. Then functional requirements are defined and the actions of each feature and the results delivered are evaluated and the mentioned **functional requirements** were achieved by the developed system.

Non-functional requirements are also considered, apart from the functional requirements. These indicate the safety and security, performance requirements and the software quality attributes that are to be expected from the final outcome. The following are the key **non-functional requirements** that are achieved from the system

- Consistency

All user interfaces follow proper guidelines and all text are grammatically correct and properly spelled out. And the interfaces are user friendly

- Availability

Finding the relevant lecture slide according to the summarized note is available for every subject and it should be searched any lecture which is uploaded by lecturers.

- Reliability

Results generated by Algorithms developed which are used to perform tasks in server are highly accurate. Also, latest configuration and technologies are used to provide high accuracy service within less time.

- Performance

The system will have to use minimum resources since some students have laptops with limited resources (RAM and CPU). It will optimize periodic tasks and routine processes to ensure that the device runs at a favorable speed at all times.

- Maintainability

Maintainability is defined as the probability of performing a successful repair action within a given time. The proposed application will be easily maintained because application is developed according to the object-oriented principals and modularization. Also, the source code will be well commented and documented for any changes or modifications done in future.

4.2.1 Analysis

The main purpose of the Slide matcher tool is to find the most similar lecture slide according to the generated summarized note. This should be basically done with text similarity and text classification. There are many technologies and functions which are developed based on text classification. In EduEasy system, the Summarized note is generated using the recorded lecture's voice. The recorded voice is converted into a text and summarized it. So the text which is in the summarized note is mainly based on human languages. So the algorithm which will be going implemented should be able to identify human languages too. So Natural Language Processing (NLP) is a good technology which deals with human languages. Get the data clean and standard format is very important for better analyzing. There are some Natural Language Processing functions which can be used to re-process the data. Using Stemming, Tokenization, removing stop words can re-process the data easily. Choosing a stop word set should be done very carefully. Because there are some stop word set which can be harmed to the meaning of the original sentence. In some sentences the words "I have" can be written as "I've". After running the removing stop word function, the word "I" would be removed and "ve" would be remained. So this will be caused to the final result of the algorithm. Therefore we should very careful to find the good stop word set.

Text similarity can be divided into mainly two parts. Lexical Similarity and Semantic similarity are the main two parts which is described about surface closeness and the meaning of the words. When we implement the algorithm to fine the most similar text, we should think about these both sub parts in text similarity. There are some algorithms which can only be identified either semantic similarity or lexical similarity or neither. Jaccard similarity algorithm is one algorithm which had to be rejected because it could not be able to identify either semantic or lexical. Jccard similarity finds the common words of two text documents which are going to be compared and produce the number of common words as the result. But it would not be the correct result. The algorithm which calculates the similarity of given documents, should be able to identify the synonyms. Because there can be synonyms in summarized note because it generated using lecture's voice. And there can be some words which can be difference from the meaning but same in words. As an example let's take the word "Leave". the word "Leave" can be taken as the leaves which are in the trees and the leaves which can be apply as medical leave or short leaves as such. Then when it finds the similarity, the Jaccard similarity cannot be able to identify the difference of this conflict.

But Gensim models can be able to identify the difference of words which is described above. It has a pre-trained data model in machine learning. So I chose genism models to implement the similarity algorithm since the accuracy of genism is very high. The Term Frequency and Inverse Document Frequency (TF-IDF) is chosen to develop the similarity calculation algorithm. TfidfVectorizer is a combined of countVectorizer and TfidfTransformer which calculate the distance of words.

Python is selected as the programming language since there are many NLP toolkits are developed for python. And spider application in Anaconda navigator is used to implement the algorithm.

4.3 Flow chart diagram

The flow chart diagram for Slide matcher tool was drawn bellow.

First the summarized not which was generated according to the recorded lecturer's voice, should be opened. The part of summarized note which has to be found from the lecture PPTs, should be selected. And then The button "Find Lecture" would be clicked. Then system finds the most similar lecture using developed similarity algorithm. And display the name of the lecture ppt. figure 3.2 is shown the flow chart diagram of slide matcher tool.

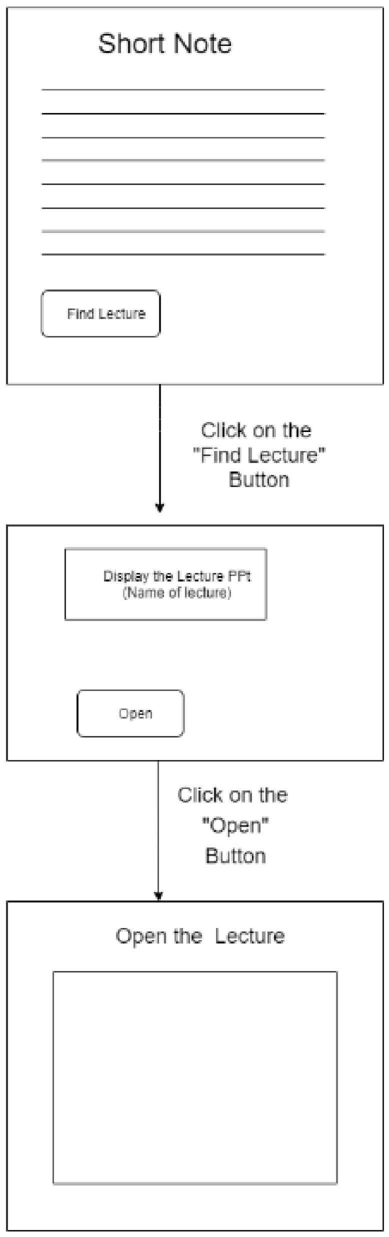

Short Note
Find Lecture
Click on the "Find Lecture" Button
Display the Lecture PPt (Name of lecture)
Open
Click on the "Open" Button
Open the Lecture

Figure 4.2 – Flow chart diagram

4.3.1Flow chart diagram for NLP functions

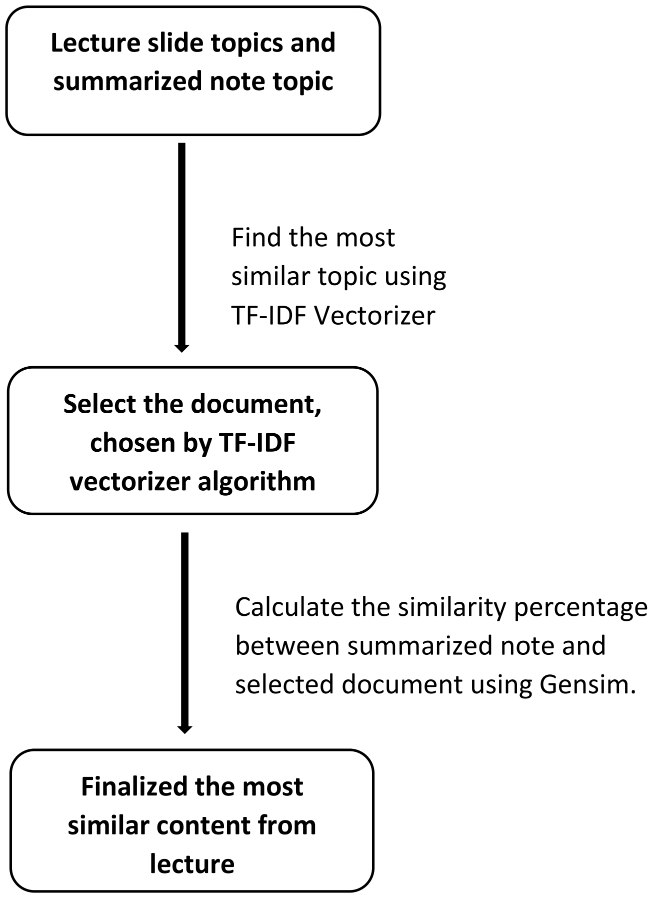
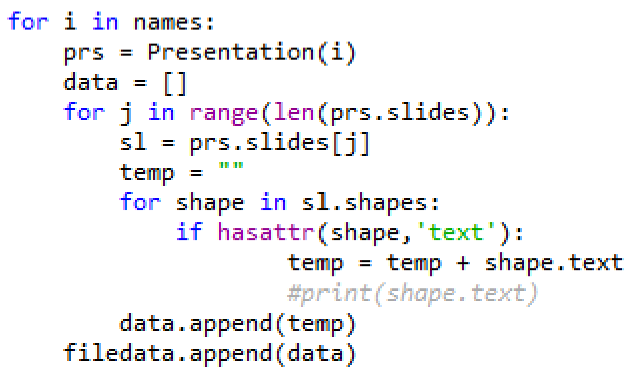

Lecture slide topics and summarized note topic
Find the most similar topic using TF-IDF Vectorizer
Select the document, chosen by TF-IDF vectorizer algorithm
Calculate the similarity percentage between summarized note and selected document using Gensim.
Finalized the most similar content from lecture

Diagram 4.1 - Flow chart of NLP functions

4.4 Implementation

4.4.1 Extract words from Power point presentation (pptx)

Word extracting is done using the presentation libraries in NLP.



```
for i in names:
    prs = Presentation(i)
    data = []
    for j in range(len(prs.slides)):
        sl = prs.slides[j]
        temp = ""
        for shape in sl.shapes:
            if hasattr(shape,'text'):
                    temp = temp + shape.text
                    #print(shape.text)
        data.append(temp)
    filedata.append(data)
```

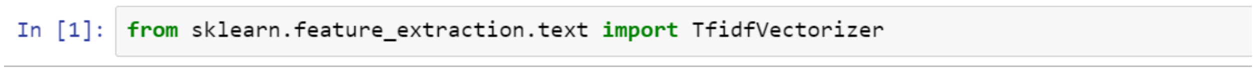
Figure 4.1 – text extraction from PPT

Figure 4.1 is shown how text is extracted from ppt. The "presentation" library should be imported. And then all the words are stored in an array.

4.4.2 Find the most similar title to the summarized note from extracted titles using TFIDF Vectorization.

- Text Extraction

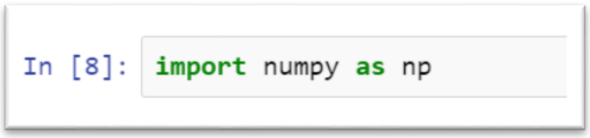
Text analysis is very important before calculating the similarity. The text needs to be extracted for analyzing purpose.



```
In [1]: from sklearn.feature_extraction.text import TfidfVectorizer
```

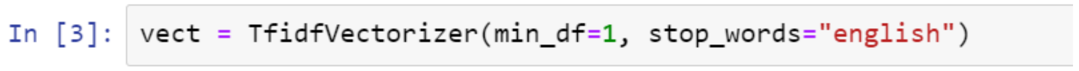
Figure 4.2 – Extracting data

The figure 4.2 is shown, sklearn.feature extraction modules are used to extract raw data from a text or an image. The sub module named sklearn.feature extraction.text.TfidfVectorizer converts raw data to a matrix for better visualization.



```
In [8]: import numpy as np
```

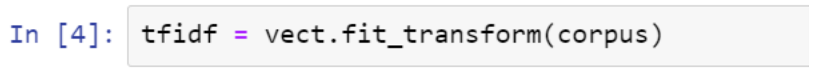
Figure 4.3 – Import "numpy"



```
In [3]: vect = TfidfVectorizer(min_df=1, stop_words="english")
```

Figure 4.4 – "TFidfVectorizer"function

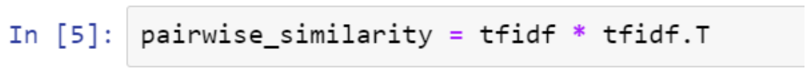
Removing stop words such as 'we', 'is', 'very' is very essential for this process. Those stop words which are represented the text can make the corpus more complex and those uninformative words should be removed. But there can be some texts such as 'I've' I the corpus. after applying the method, the word 'I've' will be tokenized into two tokens ('I' and 've') by default tokenized in CountVectorizer. Then 'I' will be removed since it is a stop word. But 've' will be remained in the corpus which can be given a wrong output. But this Vectorizer has an ability to identify these inconsistences. TfidfVectorizer is a combined of countVectorizer and TfidfTransformer.



```
In [4]: tfidf = vect.fit_transform(corpus)
```

Figure 4.5 – "fit_transform" function

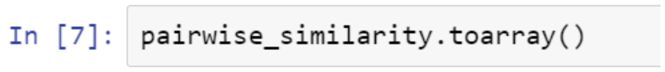
The function 'fit_transform' as shown in the figure 3.5, converts tokenized texts into numbers. All the tokens are given numbers and then it will easy for next steps. The 'fit' method computes the weight of every feature and



```
In [5]: pairwise_similarity = tfidf * tfidf.T
```

Figure 4.6 – pairwise similarity formula

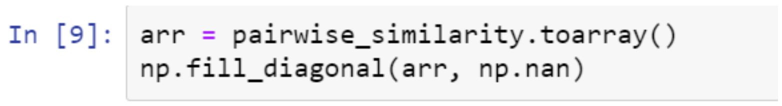
The term ‘tf’ means term frequency and ‘tf-idf’ means term frequency times inverse document frequency. Term frequency is counted as the number of times the selected term occurs in the selected document. And idf is calculated using log[(1+n)/(1+df(t)]+1 formula where the n is taken as the total number of documents in the selected set and df(t) is taken as number of documents in the selected set which has the term t. the normalization is also built in the tf-idf vectorization package as an function. As mentioned in above figure 4.6 generated tfidf matrix is multiplied by it’s transpose.



```
In [7]: pairwise_similarity.toarray()
```

Figure 3.7 – converting sparse matrix to toarray

Sparse matrix is converted into a toarray.



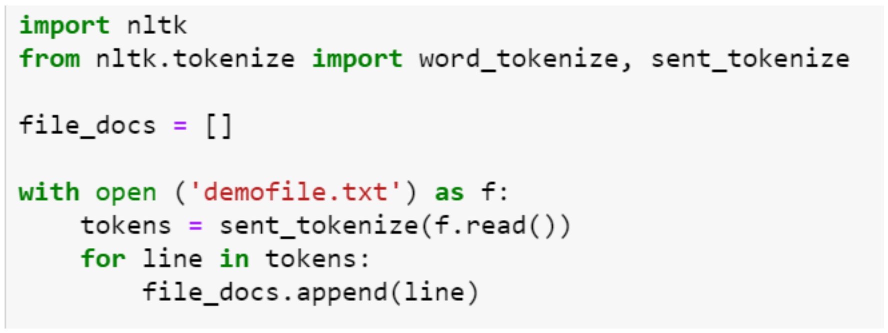
```
In [9]: arr = pairwise_similarity.toarray()
        np.fill_diagonal(arr, np.nan)
```

Figure 4.8 – “fill_diagonol” function

As shown in figure 4.8, same values for diagonal is assigned by fill_diagonal function.

4.4.3 Calculate the percentage of similarity between summarized note and selected lecture using Gensim and TF-IDF models

Two documents are taken to calculate the percentage of similarity. One document is summarized note and other one is selected lecture note. Before calculating the similarity, all the data should be re-processed.

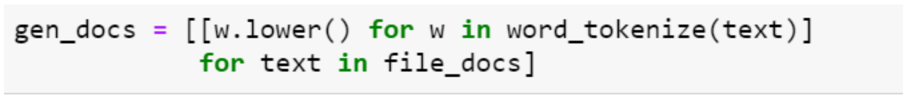


```
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize

file_docs = []

with open ('demofile.txt') as f:
    tokens = sent_tokenize(f.read())
    for line in tokens:
        file_docs.append(line)
```

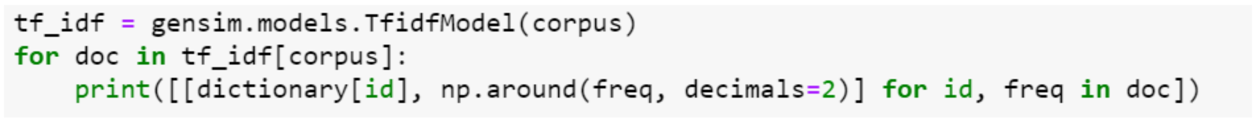
Figure 4.9 – Sentence tokenizing



```
gen_docs = [[w.lower() for w in word_tokenize(text)]
            for text in file_docs]
```

Figure 4.10 – Word tokenizing

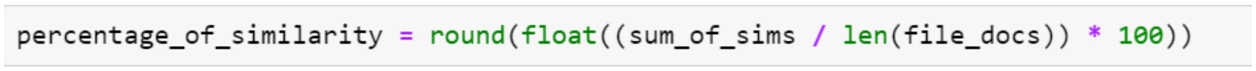
Tokenizing is a major function of Natural language tool kit (NLTK) in Natural Language processing. Figure 4.9 is shown how sentence tokenization is done. It tokenizes each and every sentence in the document. Then as shown in figure 3.10 the words in tokenized sentences are again tokenized using word_tokenize function and converted all letters into lower case letters to improve the accuracy of the results.



```
tf_idf = gensim.models.TfidfModel(corpus)
for doc in tf_idf[corpus]:
    print([[dictionary[id], np.around(freq, decimals=2)] for id, freq in doc])
```

Figure 4.11 – use of "genism models"

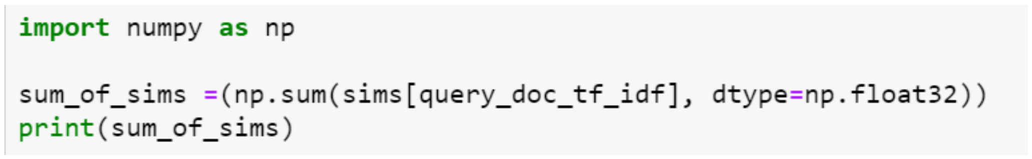
Figure 4.11 is shown the frequency according to the tokens using Tf-idf models.



```
percentage_of_similarity = round(float((sum_of_sims / len(file_docs)) * 100))
```

Figure 4.12 – calculating the percentage of similarity

figure 4.12 shown the formula which is used to calculate the percentage of similarity of selected document and summarized note. Length of the documents are specially mentioned because length can be changed from document to document. But the formula was implemented by dividing sum of the similarity of document from the length of the document which can be caused the final result. Sum of the similarity is calculated using Numpy.

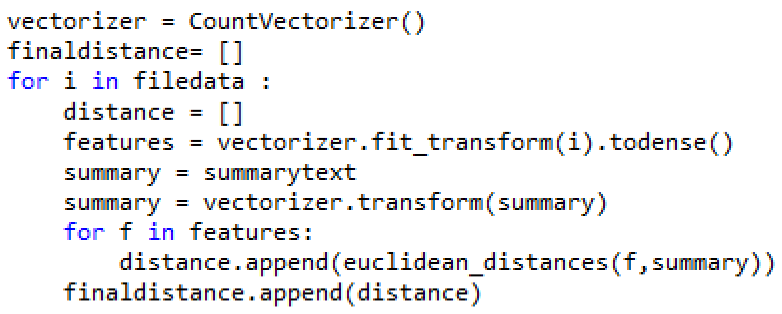


```
import numpy as np

sum_of_sims =(np.sum(sims[query_doc_tf_idf], dtype=np.float32))
print(sum_of_sims)
```

Figure 4.13 – calculating the sum of similarity

Figure 4.13 is shown how the sum of similarity is calculated using numpy.



```
vectorizer = CountVectorizer()
finaldistance= []
for i in filedata :
    distance = []
    features = vectorizer.fit_transform(i).todense()
    summary = summarytext
    summary = vectorizer.transform(summary)
    for f in features:
        distance.append(euclidean_distances(f,summary))
    finaldistance.append(distance)
```

Figure 4.14 – calculating the distance

Vector counter Pairwise technique is used to calculate the distance of words. Calculate the minimum edit distance of two words is the basic idea of this function. The “summary text” as shown in the figure 4.14, is the text which is taken from summarized note.

4.5 Commercialization aspects of the product

EduEasy E-learning web application is mainly designed for university environment. Students who wants to study smart, EduEasy system will help them to make their works easy. Students are struggling near exam to find a proper lecture notes, and references, sample questions and the course materials such as lecture PPTs. EduEasy application is a combination of those all requirements.

Slide matcher tool is developed to find the relevant lecture slides automatically. Students do not want to waste their time for finding lecture notes by manually. When the student occurs a doubt while referring the summarized note and need a better explanation, he can find the relevant lecture slide automatically.

EduEasy application is introduced with mainly three packages: Platinum, Gold and Silver. All the functionalities are included to the platinum package. Gold package is included all functions except reference finding function. And Silver package has only the lecture summarization function. And one month free trial will be given to all new users of EduEasy system.

Features provided by the EduEasy system

- A summarized note of the lecture conducted by the lecturer for each day
- Generate automatically the questions and answers for the summarized note.
- Generate automatically the reference books for additional reading.
- Navigate to the exact lecture slide from summarized note.

Benefits of EduEasy system

- Easy to find the summarized note
- Easy to find similar questions and answers
- Can find relevant references easily
- Can navigate to the relevant lecture slide automatically.

5. TESTING & IMPLEMENTATION & DISCUSSION

5.1 Testing

On slide matcher tool which has a set of model data where parameters for relevant algorithms have been changed to compare the test results so that it would be clearly visible that the algorithms we have used provide correct results.

The testing of the requirement gathering are conducted by myself. For the testing of the application unit testing had been done after implementing an independently testable portion of the tool. Alpha testing and beta testing is being conducted. A small group of students would have the application set up on their devices.

To review the features on certain functions, set of test cases will be designed. This would be a set of steps and inputs to achieve a particular goal. A set of critical and important test cases that were carried out are given below.

Test ID	01
Description	Verify the adding of summarized note content to the slide matcher tool
Steps	1. Open “Slide matcher” tool 2. Open the summarized note 3. Copy and paste the summarized note content
Expected outcome	Display the copied text
Actual outcome	Display the selected text

Table 2 Test ID 01-Verify the adding of summarized note

Test ID	02
Description	Verify uploading lecture PPT to the tool
Steps	1. Open “Slide matcher” tool 2. Click on the “Upload” button 3. Select all the lectures from popup interface.
Expected outcome	Display the selected lecture PPTs in the interface
Actual outcome	Display uploaded lecture PPTs

Table 3 Test ID 02- verify uploading lecture

Test ID	03
Description	Verify the search function
Steps	1. Open “Slide matcher” tool 2. Click on the “search” button
Expected outcome	Display the PPT name.
Actual outcome	Display the most similar lecture PPT along with the name.

Table 3 Test ID 02- verify searching

5.2 Implementation

This section of the document will explain the technologies that were used to implement each functionalities of the proposed system.

Python is a programming language which is easy to learn which contains user friendly data structures and provides a large standard library so that it will help to reduce the length of code written. So we have implemented the optimization algorithms and prediction algorithms using Spyder application under Anaconda Navigator which is a free and open source distribution of the python programming language.

5.3 Experimental Setup

I have implemented the optimization algorithms using Spyder application under Anaconda Navigator which is a free and open source distribution of the python programming language. Experiments were performed on a laptop with the following properties;

- CPU: Intel Core i7 (2.40GHz)

- RAM: 8GB
- OS: Windows 10 (64-bit Operating System)

5.4. Results and discussions

5.4.1 Results

The basic part of Slide matcher component is find the most relevant from the database lecture according to the summarized note. Finding the similarity is done in two ways.

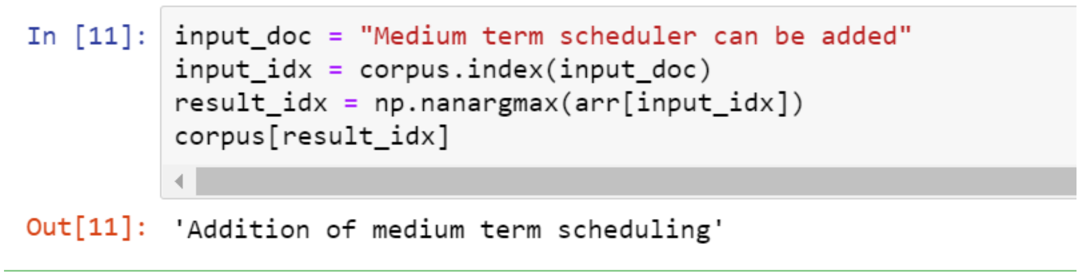
Comparing extracted topics and summarized note's topic using TF-IDF Vectorization

The inputs are extracted topics as follows.

- "CPU Switch from Process to Process"
- "Process Scheduling"
- "Representation of Process scheduling"
- "Representation of Process scheduling"
- "Addition of medium term scheduling"

"Medium term scheduler can be added***"*** is taken as the selected summarized note phrase which has many similar words in extracted topics with changes of verb form and suffixes. The word "Addition" which is in the extracted topic list, has been changed into "Addition" in the summarized note phrase which was selected for the comparison using developed algorithm. Same as Addition, there is an another word: "scheduling" which has been changed its suffixes to "Scheduler" in the summarized note phrase. So TF-IDF Normalizes those words and no need to normalize them manually.

After running the TF-IDF algorithm, "***Addition of medium term scheduling***" has displayed as the result, shown in figure 5.1.



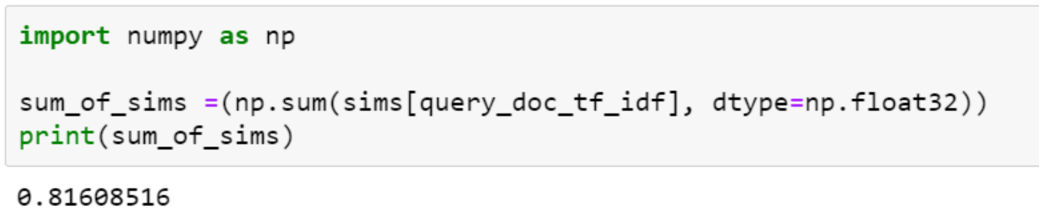
```
In [11]: input_doc = "Medium term scheduler can be added"
         input_idx = corpus.index(input_doc)
         result_idx = np.nanargmax(arr[input_idx])
         corpus[result_idx]

Out[11]: 'Addition of medium term scheduling'
```

Figure 5.1 – Output of topic analyzing

Calculate the percentage of similarity between summarized note and selected lecture.

Calculate the sum of similarity is done as shown in figure 2. It produces the sum of the similarity value. Inputs are summarized note content and lecture content.



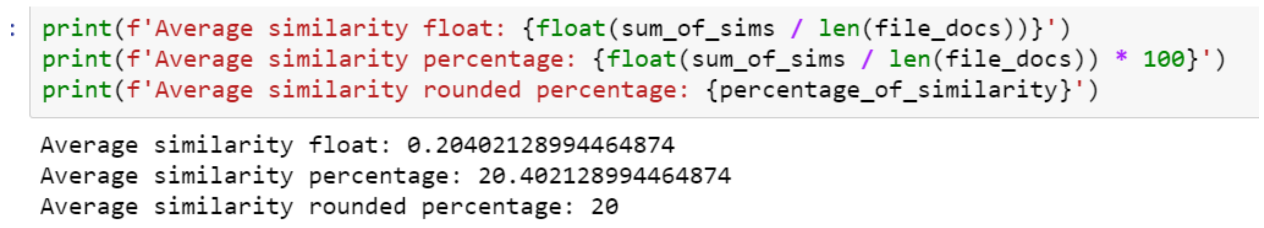
```
import numpy as np

sum_of_sims =(np.sum(sims[query_doc_tf_idf], dtype=np.float32))
print(sum_of_sims)
```

```
0.81608516
```

Figure 5.2 – calculate the sum of similarity

After running the algorithm developed using Gensim, calculated similarity percentage is displayed as shown in figure 5.2.

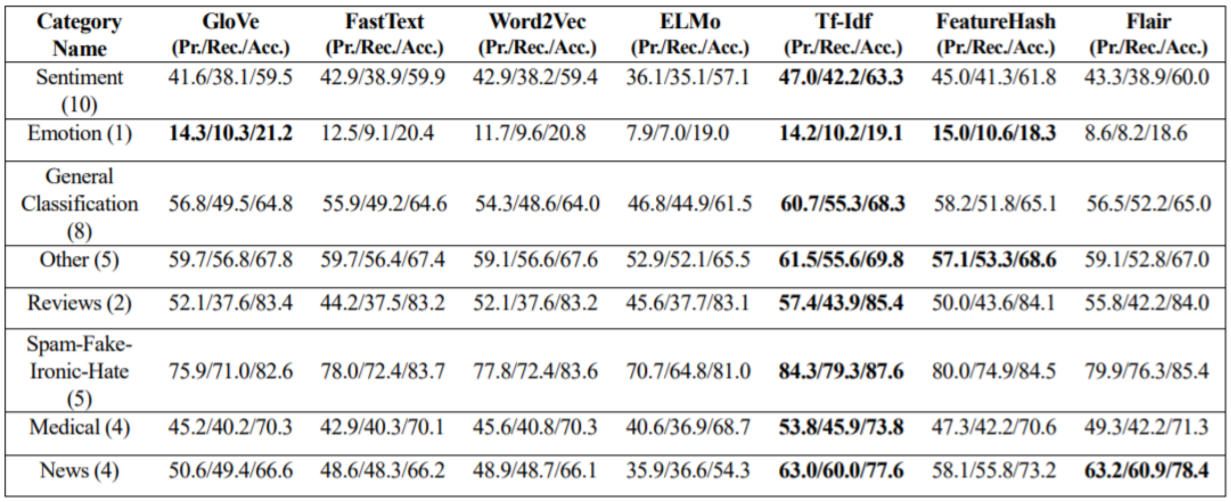


```
: print(f'Average similarity float: {float(sum_of_sims / len(file_docs))}')
  print(f'Average similarity percentage: {float(sum_of_sims / len(file_docs)) * 100}')
  print(f'Average similarity rounded percentage: {percentage_of_similarity}')
```

```
Average similarity float: 0.20402128994464874
Average similarity percentage: 20.402128994464874
Average similarity rounded percentage: 20
```

5.4.2 Accuracy of the test results.

TF-IDF vectorization models and Gensim models are used to implement the algorithm which calculates the similarity of text documents. Used text classification models should have a good performance such as accuracy, precision and recall.



Category Name	GloVe (Pr./Rec./Acc.)	FastText (Pr./Rec./Acc.)	Word2Vec (Pr./Rec./Acc.)	ELMo (Pr./Rec./Acc.)	Tf-Idf (Pr./Rec./Acc.)	FeatureHash (Pr./Rec./Acc.)	Flair (Pr./Rec./Acc.)
Sentiment (10)	41.6/38.1/59.5	42.9/38.9/59.9	42.9/38.2/59.4	36.1/35.1/57.1	**47.0/42.2/63.3**	45.0/41.3/61.8	43.3/38.9/60.0
Emotion (1)	**14.3/10.3/21.2**	12.5/9.1/20.4	11.7/9.6/20.8	7.9/7.0/19.0	**14.2/10.2/19.1**	**15.0/10.6/18.3**	8.6/8.2/18.6
General Classification (8)	56.8/49.5/64.8	55.9/49.2/64.6	54.3/48.6/64.0	46.8/44.9/61.5	**60.7/55.3/68.3**	58.2/51.8/65.1	56.5/52.2/65.0
Other (5)	59.7/56.8/67.8	59.7/56.4/67.4	59.1/56.6/67.6	52.9/52.1/65.5	**61.5/55.6/69.8**	**57.1/53.3/68.6**	59.1/52.8/67.0
Reviews (2)	52.1/37.6/83.4	44.2/37.5/83.2	52.1/37.6/83.2	45.6/37.7/83.1	**57.4/43.9/85.4**	50.0/43.6/84.1	55.8/42.2/84.0
Spam-Fake-Ironic-Hate (5)	75.9/71.0/82.6	78.0/72.4/83.7	77.8/72.4/83.6	70.7/64.8/81.0	**84.3/79.3/87.6**	80.0/74.9/84.5	79.9/76.3/85.4
Medical (4)	45.2/40.2/70.3	42.9/40.3/70.1	45.6/40.8/70.3	40.6/36.9/68.7	**53.8/45.9/73.8**	47.3/42.2/70.6	49.3/42.2/71.3
News (4)	50.6/49.4/66.6	48.6/48.3/66.2	48.9/48.7/66.1	35.9/36.6/54.3	**63.0/60.0/77.6**	58.1/55.8/73.2	**63.2/60.9/78.4**

Table 2. Mean Performance Metrics on a category basis for all Vectorizers (dataset size less than 10K)

Figure 5.3 – comparison the performance of models

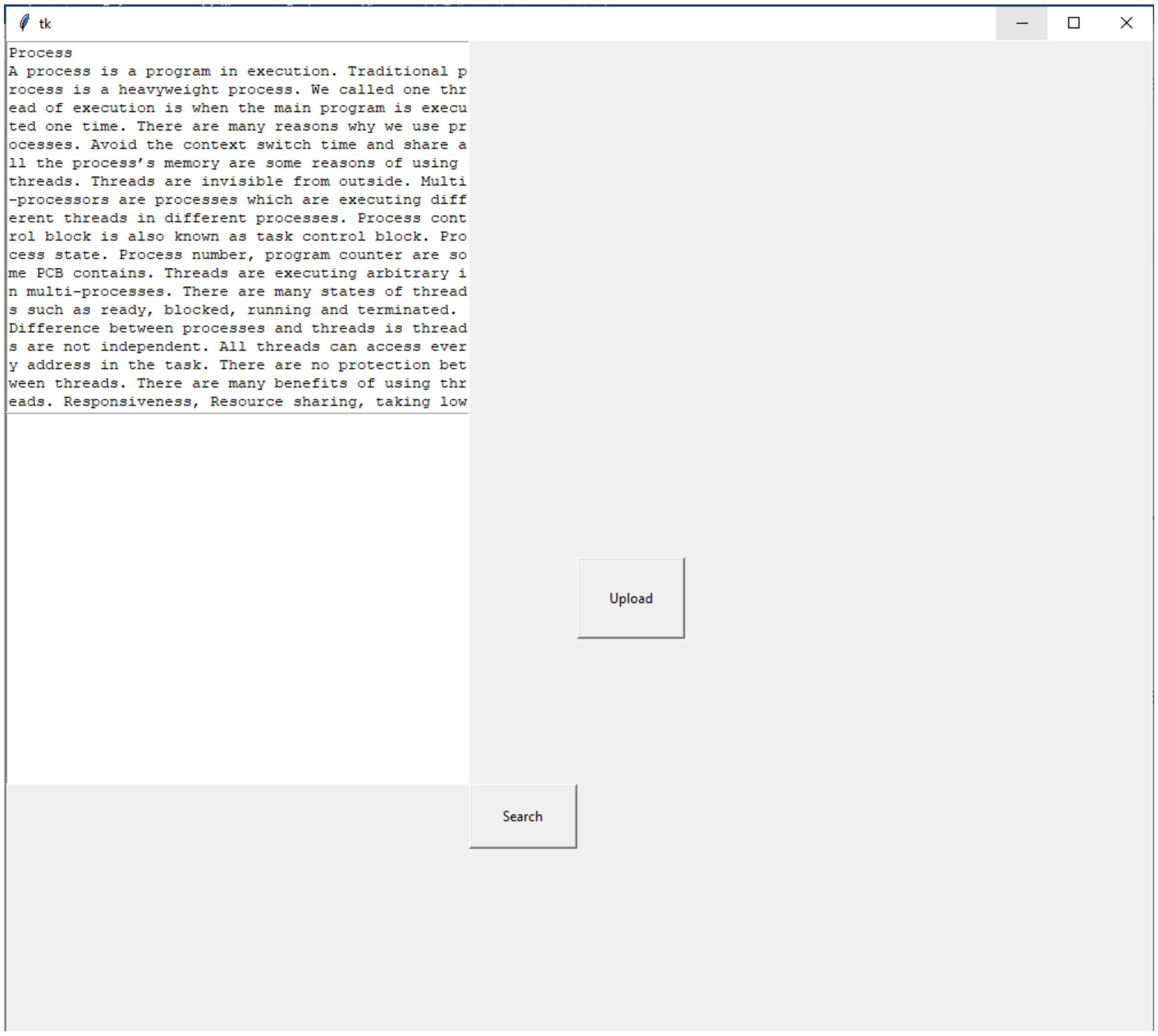
As shown in the figure 4.3, TF-idf models has the highest number for accuracy. This is taken from a research conducted about Large Scale Systematic Comparison of Count-Based and Prediction-Based Vectorizers for Text Classification [6].

What can be done to improve the accuracy of similarity?

Removing hyper links, Changing the case to lower case and removing typo words can make the performance and accuracy of the results high.

5.4.3 Output of the Slide matcher tool

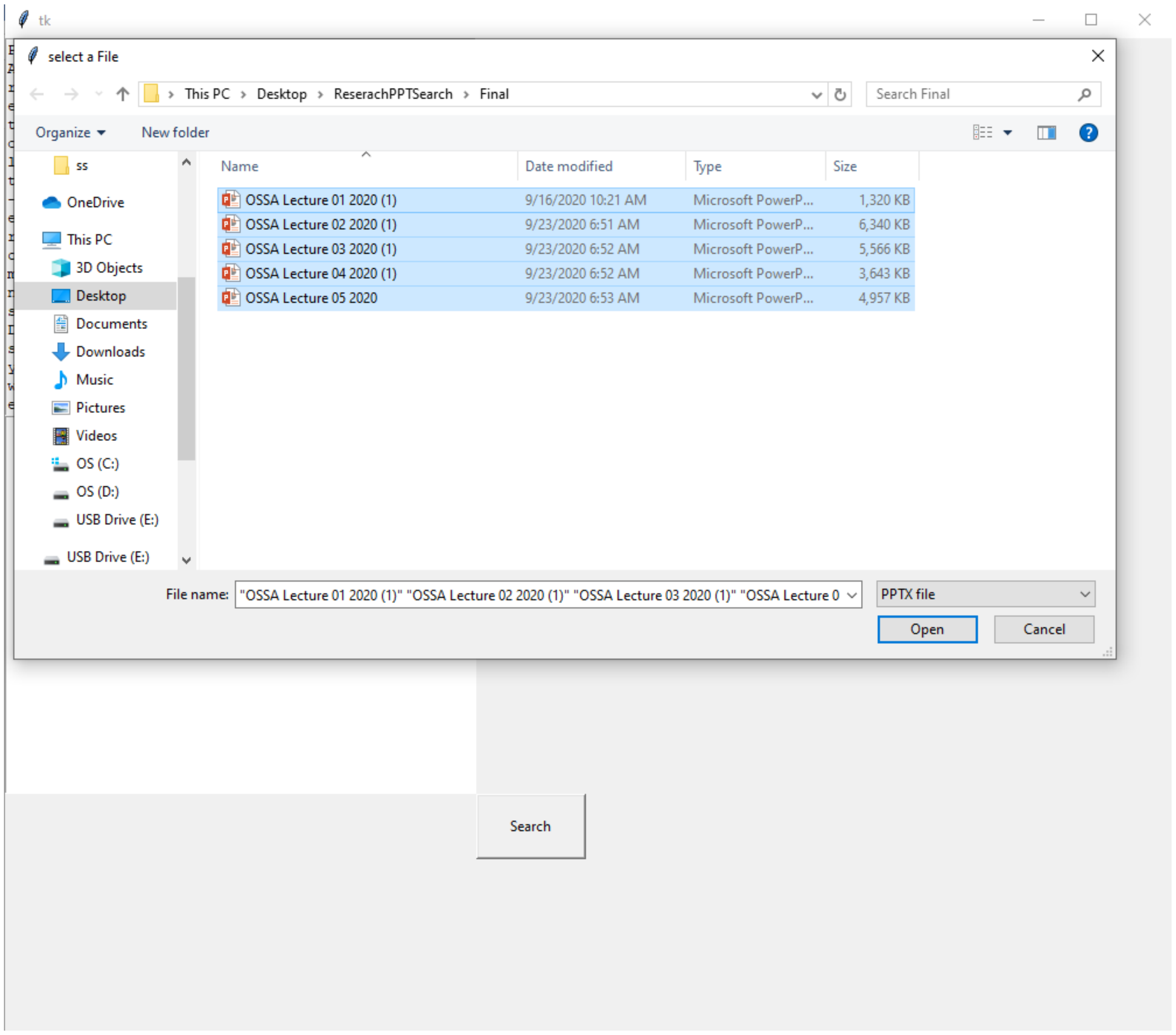
Uploading the summarized note content which has to be copied from summarized note. User can copy whole content or part of the note.



```
Process
A process is a program in execution. Traditional p
rocess is a heavyweight process. We called one thr
ead of execution is when the main program is execu
ted one time. There are many reasons why we use pr
ocesses. Avoid the context switch time and share a
ll the process’s memory are some reasons of using
threads. Threads are invisible from outside. Multi
-processors are processes which are executing diff
erent threads in different processes. Process cont
rol block is also known as task control block. Pro
cess state. Process number, program counter are so
me PCB contains. Threads are executing arbitrary i
n multi-processes. There are many states of thread
s such as ready, blocked, running and terminated.
Difference between processes and threads is thread
s are not independent. All threads can access ever
y address in the task. There are no protection bet
ween threads. There are many benefits of using thr
eads. Responsiveness, Resource sharing, taking low
```

Figure 5.4 – Interface 1

When the user clicks on the upload button, this interface is popped up. This is about uploading the all lecture PPTs to the system. Then all the PPTs should be selected and click the “open” button.



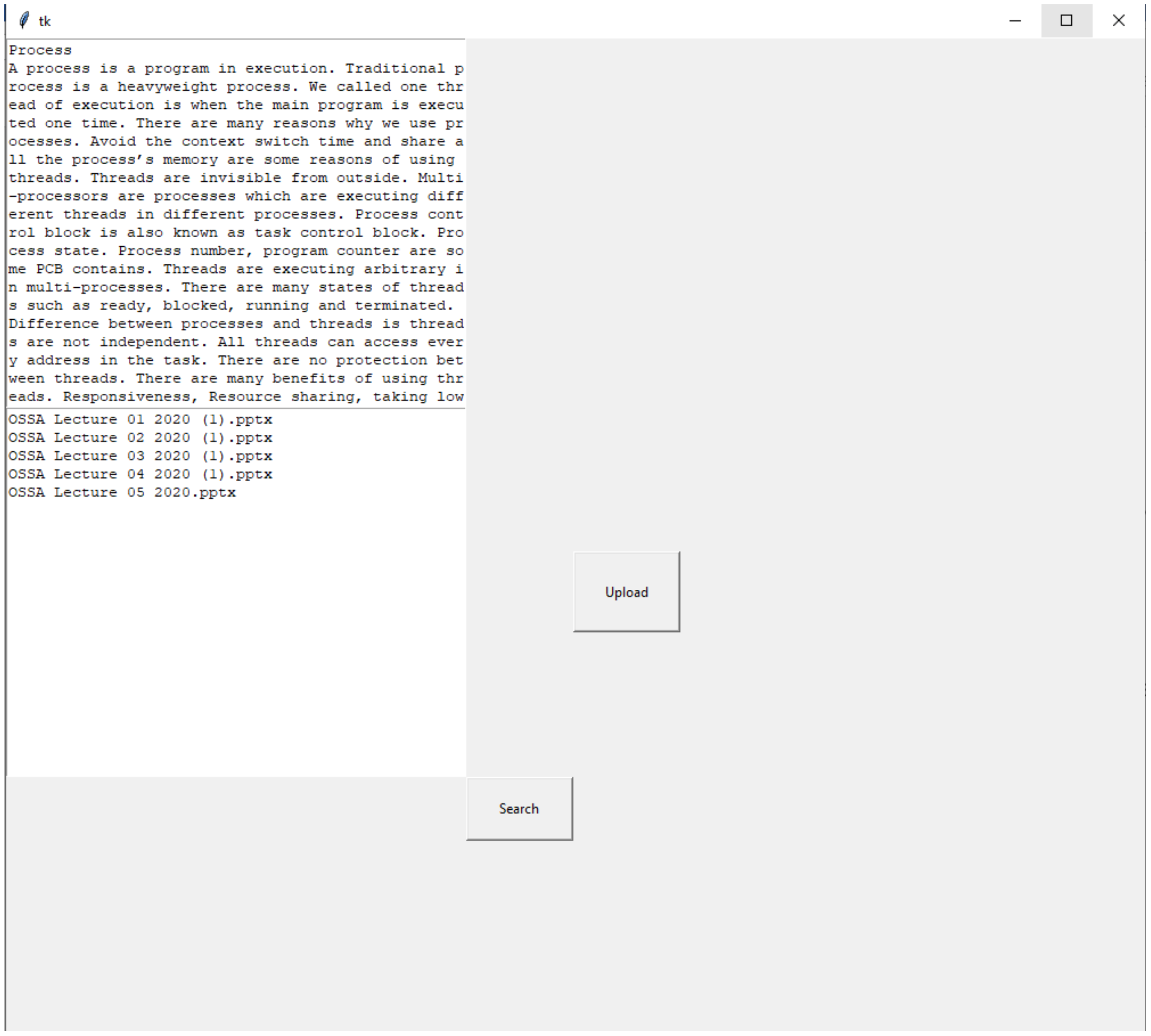
Name	Date modified	Type	Size
OSSA Lecture 01 2020 (1)	9/16/2020 10:21 AM	Microsoft PowerP...	1,320 KB
OSSA Lecture 02 2020 (1)	9/23/2020 6:51 AM	Microsoft PowerP...	6,340 KB
OSSA Lecture 03 2020 (1)	9/23/2020 6:52 AM	Microsoft PowerP...	5,566 KB
OSSA Lecture 04 2020 (1)	9/23/2020 6:52 AM	Microsoft PowerP...	3,643 KB
OSSA Lecture 05 2020	9/23/2020 6:53 AM	Microsoft PowerP...	4,957 KB

File name: "OSSA Lecture 01 2020 (1)" "OSSA Lecture 02 2020 (1)" "OSSA Lecture 03 2020 (1)" "OSSA Lecture 0

PPTX file

Figure 5.5 – Interface 2

Then selected lecture PPTs are displayed. After that user can click on the “search” button to find the most similar lecture for the selected summarized note.

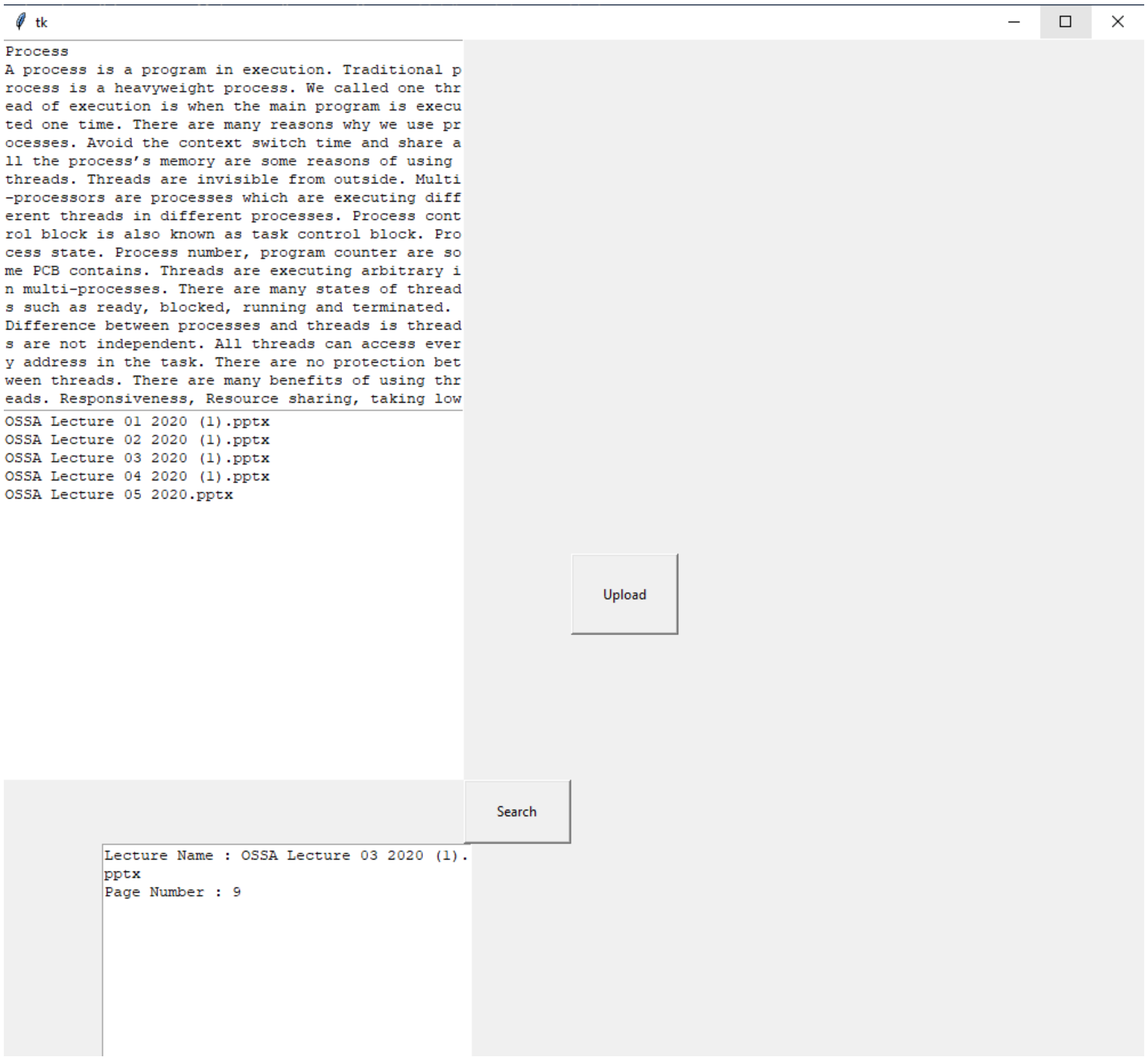


```
Process
A process is a program in execution. Traditional p
rocess is a heavyweight process. We called one thr
ead of execution is when the main program is execu
ted one time. There are many reasons why we use pr
ocesses. Avoid the context switch time and share a
ll the process’s memory are some reasons of using
threads. Threads are invisible from outside. Multi
-processors are processes which are executing diff
erent threads in different processes. Process cont
rol block is also known as task control block. Pro
cess state. Process number, program counter are so
me PCB contains. Threads are executing arbitrary i
n multi-processes. There are many states of thread
s such as ready, blocked, running and terminated.
Difference between processes and threads is thread
s are not independent. All threads can access ever
y address in the task. There are no protection bet
ween threads. There are many benefits of using thr
eads. Responsiveness, Resource sharing, taking low
```

```
OSSA Lecture 01 2020 (1).pptx
OSSA Lecture 02 2020 (1).pptx
OSSA Lecture 03 2020 (1).pptx
OSSA Lecture 04 2020 (1).pptx
OSSA Lecture 05 2020.pptx
```

Figure 5.6 – Interface 3

The most similar lecture slide is displayed as the result.



```
Process
A process is a program in execution. Traditional p
rocess is a heavyweight process. We called one thr
ead of execution is when the main program is execu
ted one time. There are many reasons why we use pr
ocesses. Avoid the context switch time and share a
ll the process's memory are some reasons of using
threads. Threads are invisible from outside. Multi
-processors are processes which are executing diff
erent threads in different processes. Process cont
rol block is also known as task control block. Pro
cess state. Process number, program counter are so
me PCB contains. Threads are executing arbitrary i
n multi-processes. There are many states of thread
s such as ready, blocked, running and terminated.
Difference between processes and threads is thread
s are not independent. All threads can access ever
y address in the task. There are no protection bet
ween threads. There are many benefits of using thr
eads. Responsiveness, Resource sharing, taking low
```

```
OSSA Lecture 01 2020 (1).pptx
OSSA Lecture 02 2020 (1).pptx
OSSA Lecture 03 2020 (1).pptx
OSSA Lecture 04 2020 (1).pptx
OSSA Lecture 05 2020.pptx
```

```
Lecture Name : OSSA Lecture 03 2020 (1).
pptx
Page Number : 9
```

Figure 5.7 – Interface 4

5.4.4 Research Findings and Discussion

Comparing the text similarity can be done in many ways. Matching the exact words of two documents cannot be taken as the real output. Because there are many things that has to be done before comparing two documents.

- Can get the real output using String matching algorithm?
  String matching algorithms only match the number of words which has the same letters. But in a summarized note which is converted lecturer’s voice into text, cannot find the exact similar words to the lecture PPT. because lecturer can use various words when he is conducting the lecture. So an algorithm or a technology which can identify human languages, should be used to find the similarity of this kind of documents.

- Use of NLP functions.
  Tokenization, stop words, stemming, lemmatization are some functions in NLP which help to re-process the data. Get the data cleaned and standard format is very important. Then the words with suffixes, prefixes, with different tenses are changed into the general form. The lecturer can use various words to explain the lesson which are converted into text.

- Calculate the similarity.
  Calculate the similarity of given documents are done with TF-IDF vectorization methods and gensim models. These functions can be able to identify the synonyms other problems discussed in the methodology section.

6. CONCLUSION

I have proposed a tool to find the most similar lecture according to the generated summarized note. Students who are using EduEasy web application can use the Slide Matcher tool to find the relevant lecture slide while referring the summarized note which was created by converting recorded lecturer's voice.

I have developed an algorithm to calculate the percentage of similarity of the chosen documents and find the most similar lecture PPT using extracted words of PPTs. I used TF-IDF count vectorization methods, gensim and vector counter pairwise technique.

The finalized product of the research is to provide a better experience in E-Learning. Users can select the summarized note and upload the lecture PPTs to the Slide Matcher tool. It also provides the following benefits;

- Ability to upload the lecture PPTs to the tool.
- Can copy the summarized note content and paste it on the Slide Matcher tool. This content can be either the whole summarized note or the sub section of the note.
- Automatically display the most relevant lecture slide according to the selected summarized note.

The current tool "Slide matcher" already achieved the above mentioned benefits and the proposed "EduEasy" system also achieved all the requirements.

7. REFERENCES

[1] LajanugenLogeswaran, HonglakLee, DragomirRadev "Sentence Orderingand Coherence Modeling Using Recurrent Neural Networks" Department of Computer Science & Engineering, University of Michigan 2Department of Computer Science, Yale University, The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)

[2] Pantulkar Shravanthi, Dr. B. Srinivasu "Semantic similarity between sentences" M.tech Scholar Dept. of Computer Science and Engineering, Stanley College of Engineering and Technology for Women, Telangana- Hyderabad, India, International Research Journal of Engineering and Technology

[3] Adrien Sieg "Text Similarities : Estimate the degree of similarity between two texts". Available (https://medium.com/@adriensieg/text-similarities-da019229c894)

[4] Xiaohao Yang, Jia Liu, "Semi-supervised learning of dialogue acts using sentence similarity based on word embeddings" department of electronic engineering, Tsighua university Beijing100084, China.

[5] Rupak Chakraborty, Ashima Elhence and Kapil Arora, "Sparse Victory – A Large Scale Systematic Comparison of Count-Based and Prediction-Based Vectorizers for Text Classification," proceedings of the International Conference on Recent Advances in Natural Language Processing(RANLP), Varna, Bulgaria, Sep. 2019, pp. R19-1022

[6] Omid Shashmirzadi, Adam Lugowski and Kenneth Younge, "Text Similarity in Vector Space Models. AComparative Study," IEEE International Conference on Machine Learning and Application (ICMLA), Boca Raton, FL, USA, USA, Dec 2019 pp, 19378206