A

PROJECT REPORT

ON

**File Resemblance Analyzer**

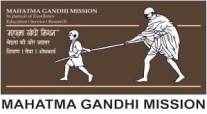
Submitted in partial fulfilment of the requirements Of University of Mumbai for the degree of

**Bachelor of Engineering (BE Sem-VIII)** By:

**Akash Laxman Hire (116CP1131A) Ameya Gulhane (114CP1359A) Bharat Dussa (116CP1291A) Aditya Ajay Choure (116CP1043A)**

Under the Supervision of

**Prof. Abhijit Patil.**

****

Department of Computer Engineering

Mahatma Gandhi Mission’s College of Engineering &TechnologyKamothe,Navi Mumbai-400209

I

**University of Mumbai**

**ACADEMIC YEAR 2019-20**

**Project Report for Bachelor of Engineering**

The Project report Entitled **“File Resemblance Analyzer”** by**, AkashLaxmanHire, Ameya Gulhane, Bharat Dussa, Aditya Ajay Choure** is approval forthe degree of **“Bachelor of Engineering in Computer EngineeringBE( SEMVIII)”**

**Examiners:**

**1. ………………………………………**

**2. ………………………………………**

**Date:**

**Place:**

II

**Declaration**

We declare that this written submission represents our ideas in our own words andwhereothers' ideas or words have been included, we have adequately cited and referencedtheoriginal sources. We also declare that we have adhered to all principles of academichonestyand integrity and have not misrepresented or fabricated or falsified any idea/data/fact/sourcein my submission. We understand that any violation of the above will be causefordisciplinary action by the Institute and can also evoke penal action fromthe sources whichhave thus not been properly cited or from whom proper permission has not been takenwhenneeded.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Akash Hire**

**Ameya Gulhane**

**Aditya Choure**

**Bharat Dussa**

III

**Date:**

**Table of Content**

| SR.NO. | TOPIC | PG. NO |
| --- | --- | --- |
| 1. | Introduction  1.1 Introduction  1.2 Problem Statement | 6 |
| 2. | Review Of Literature  2.1 Django  2.2 NLTK  2.3 Modules | 7 |
| 3. | Architecture | 12 |
| 4. | Source code | 10 |
| 5. | Output | 21 |
| 6. | Applications | 24 |
| 7. | Future Scope | 25 |
| 8. | Conclusion | 26 |
| 9. | Reference | 27 |

IV

**Table of Figure**

| **SR.NO** | **FIGURE NAME** | **PG. NO.** |
| --- | --- | --- |
| 3.1 | System Architecture | 12 |
| 5.1 | Main Page | 21 |
| 5.2 | Data Entry | 21 |
| 5.3 | Uploading File 1 | 22 |
| 5.4 | Uploading File 2 | 22 |
| 5.5 | Result Display using GUI | 23 |

V

**ABSTRACT**

*In today's area of internet and online services, data is generating at incredible speedandamount. Generally, Data analyst, engineer, and scientists are handling relational or tabulardata. These tabular data columns have either numerical or categorical data. Generateddatahas a variety of structures such as text, image, audio, and video. Online activities suchasarticles, website text, blog posts, social media posts are generating unstructured textual data.Corporate and business need to analyze textual data to understand customer activities,opinion, and feedback to successfully derive their business. To compete with big textual data,text analytics is evolving at a faster rate than ever before.*

6

**CHAPTER -1**

**INTRODUCTION**

**1.1. INTRODUCTION**

Text Analytics has lots of applications in today's online world. By analyzingtweetsonTwitter, we can find trending news and peoples reaction on a particular event. Amazoncanunderstand user feedback or review on the specific product. BookMyShowcandiscoverpeople's opinion about the movie. Youtube can also analyze and understandpeoplesviewpoints on a video.

**1.2. PROBLEM STATEMENT:**

Document similarities are measured based on the content overlap between documents. Withthe large number of text documents in our life, there is a need toautomatically processthosedocuments for information extraction, similarity clustering, and search applications. Thereexist a vast number of complex algorithms to solve this problem. One of such algorithmsisa cosine similarity - a vector based similarity measure. The cosine distanceof twodocuments is defined by the angle between their feature vectors which are, in our case, wordfrequency vectors. The word frequency distribution of a document is a mapping fromwordsto their frequency count.NLTK is a powerful Python package that provides a set of diversenatural languages algorithms. It is free, opensource, easy to use, large community, andwelldocumented.

7

**CHAPTER -2**

**REVIEWOF LITERATURE**

**2.1. Django :**

Django is an MVT web framework used to build web applications. It defines itself asa“batteries included” web framework, with robustness and simplicity to help webdeveloperswrite clean, efficient and powerful code. It is among the most famous web frameworksoutthere in the world and it’s one of the most used frameworks as well. It’s used byInstagram,Youtube, Google and even NASA for their website. So let’s break it down evenfurthertolearn more about it

The template layer is used to separate the data from the way it’s actually presentedandviewed by the user. The template layer is similar to the MVC’s View layer. If you’refamiliarwith templating in other languages, it’s kind of the same in Django; you use an HTMLlikesyntax that is later compiled to HTML with all the respective data injected. Of course, thereare formats for templates other than HTML, if you want to generate XMLdocumentsorJSON files, etc.The mechanism for creating a simple GUI application on tkinter.

**2.2 NLTK :**

NLTK is a leading platform for building Python programs to work with human languagedata.It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet,along with a suite of text processing libraries for classification, tokenization, stemming,tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLPlibraries, andan active discussion forum.NLTK is suitable for linguists, engineers, students, educators,researchers, and industry users alike. NLTK is available for Windows, Mac OSX, andLinux.Best of all, NLTK is a free, open source, community-driven project.

The context in which selenium web driver comes into action is as follows:-

NLTK stands for Natural Language Toolkit. This toolkit is one of the most powerful NLPlibraries which contains packages to make machines understand human language andreplytoit with an appropriate response.

8

**2.3 Gensim :**

Gensim is an open-source library for unsupervised topic modeling and natural languageprocessing, using modern statistical machine learning.

Gensim is implemented in Python and Cython. Gensim is designed to handle largetextcollections using data streaming and incremental online algorithms, which differentiatesitfrom most other machine learning software packages that target only in-memory processing.

*Gensim = “Generate Similar”* is a popular open source natural language processinglibraryused for unsupervised topic modeling. It uses top academic models and modernstatisticalmachine learning to perform various complex tasks such as Building document or wordvectors, Corpora, performing topic identification, performing document comparison(retrieving semantically similar documents), analysing plain-text documents for semanticstructure.To create a dictionary, we need a list of words from our text (also knownas tokens).In the following line, we split our document into sentences and then the sentences intowords.

**Installing Gensim**

If you use pip installer to install your Python libraries, you can use the following commandtodownload the Gensim library:

$ pip install gensim

9

**Tokenization of words:**

We use the method word\_tokenize() to split a sentence into words. The output of wordtokenization can be converted to Data Frame for better text understanding in machinelearningapplications. It can also be provided as input for further text cleaning steps such as punctuation removal, numeric character removal or stemming. Machine learning modelsneednumeric data to be trained and make a prediction. Word tokenization becomes a crucial part of the text (string) to numeric data conversion. Please read about Bag of Words or CountVectorizer. Please refer to below example to understand the theory better.

**Creating Dictionaries:**

Statistical algorithms work with numbers, however, natural languages contain data intheformof text. Therefore, a mechanism is needed to convert words to numbers. Similarly, after applying different types of processes on the numbers, we need to convert numbers backtotext.

One way to achieve this type of functionality is to create a dictionary that assigns a numericID to every unique word in the document. The dictionary can then be used to find the numericequivalent of a word and vice versa.

**Creating Dictionaries using In-Memory Objects**

It is super easy to create dictionaries that map words to IDs using Python's Gensimlibrary.Wefirst import the gensim library along with the corpora module from the library. Next, wehavesome text (which is the first part of the first paragraph of the Wikipedia article on Artificial Intelligence) stored in the text variable.To create a dictionary, we need a list of words fromour text (also known as tokens). In the following line, we split our document into sentencesand then the sentences into words.

tokens = [[token for token in sentence.split()] for sentence in text]

We are now ready to create our dictionary. To do so, we can use the Dictionary object of the corpora module and pass it the list of tokens.

10

**2.3 Modules:**

∙ **Data Entry Window:**

User can enter Files to search for resemblance in this Window. This Data includes-

1. File to be analyzed

2. File to be resembled for similarity

3. Result

.

∙ **Result Window:**

Shows Graphical meter after analyzing the similarity of two files in percentagemeterusing GUI

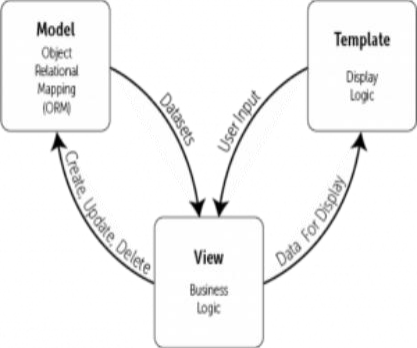
∙ **Result Save:**

The Contents of Result Window can be exported as .csv file.

11

**CHAPTER - 3**

**ARCHITECTURE**

**Fig 3.1 System Architecture**

12

**CHAPTER - 4**

**SOURCE CODE**

4.1 View

from django.shortcuts import render, redirect, get\_object\_or\_404 import gensim

import nltk

import numpy as np

from nltk.tokenize import word\_tokenize, sent\_tokenize

from .models import Document

from .forms import DocumentForm

def document\_upload(request):

documents = Document.objects.order\_by('-id')

if request.method == 'POST':

form = DocumentForm(request.POST, request.FILES)

if form.is\_valid():

form.save()

return redirect('home')

else:

form = DocumentForm()

return render(request, 'document\_upload.html', {

'form': form,

'documents':documents,

})

def similarity(request, id):

document = get\_object\_or\_404(Document, id=id)

file\_docs = []

file2\_docs = []

avg\_sims = []

with open ('media/' + document.document.name) as f:

tokens = sent\_tokenize(f.read())

for line in tokens:

file\_docs.append(line)

13

length\_doc1 = len(file\_docs)

gen\_docs = [[w.lower() for w in word\_tokenize(text)] for text in file\_docs]

dictionary = gensim.corpora.Dictionary(gen\_docs)

corpus = [dictionary.doc2bow(gen\_doc) for gen\_doc in gen\_docs] tf\_idf = gensim.models.TfidfModel(corpus)

sims = gensim.similarities.Similarity('workdir/',tf\_idf[corpus], num\_features=len(dictionary))

with open ('media/' + document.document2.name) as f: tokens = sent\_tokenize(f.read())

for line in tokens:

file2\_docs.append(line)

for line in file2\_docs:

query\_doc = [w.lower() for w in word\_tokenize(line)] query\_doc\_bow = dictionary.doc2bow(query\_doc) query\_doc\_tf\_idf = tf\_idf[query\_doc\_bow]

print('Comparing Result:', sims[query\_doc\_tf\_idf]) sum\_of\_sims =(np.sum(sims[query\_doc\_tf\_idf], dtype=np.float32)) avg = sum\_of\_sims / len(file\_docs)

print(f'avg: {sum\_of\_sims / len(file\_docs)}')

avg\_sims.append(avg)

total\_avg = np.sum(avg\_sims, dtype=np.float)

print(total\_avg)

percentage\_of\_similarity = round(float(total\_avg) \* 100) if percentage\_of\_similarity >= 100:

percentage\_of\_similarity = 100

return render(request, 'document.html', {

'percentage\_of\_similarity':percentage\_of\_similarity,

})

14

**4.2 Form**

from django import forms

from .models import Document

class DocumentForm(forms.ModelForm):

class Meta:

model = Document

fields = ('description', 'document', 'document2', )

@import url('https://fonts.googleapis.com/css?family=Baloo&display=swap');

body{

font-family: 'Baloo', cursive;

}

/\* home css \*/

.file {

position: relative;

}

.file label {

background: #39D2B4;

padding: 5px 20px;

color: #fff;

font-weight: bold;

font-size: .9em;

transition: all .4s;

}

.file input {

position: absolute;

15

display: inline-block;

left: 0;

top: 0;

opacity: 0.01;

cursor: pointer;

}

.file input:hover + label, .file input:focus + label { background: #34495E; color: #39D2B4;

}

form { width: 225px;

margin: 30 auto;

text-align:center;

}

a{

color:black;

}

a:hover{

color: #39D2B4;

}

h1, h2 {

margin-bottom: 5px;

font-weight: normal;

text-align: center;

color:#aaa;

16

}

h2 {

margin: 5px 0 2em;

color: #39D2B4;

}

h2 + P {

text-align: center;

}

input[type="text"]::placeholder {

/\* Firefox, Chrome, Opera \*/ text-align: center;

}

.txtcenter {

margin-top: 4em;

font-size: .9em;

text-align: center;

color: #aaa;

}

.copy {

margin-top: 2em;

}

.copy a {

text-decoration: none;

color: #39D2B4;

}

/\* button \*/

.btn {

17

border-radius: 5px;

padding: 15px 25px;

font-size: 22px;

text-decoration: none;

margin: 20px;

color: #fff;

position: relative;

display: inline-block;

}

.btn:active {

transform: translate(0px, 5px);

-webkit-transform: translate(0px, 5px); box-shadow: 0px 1px 0px 0px;

}

.blue {

background-color: #55acee;

box-shadow: 0px 5px 0px 0px #3C93D5; }

.blue:hover {

background-color: #6FC6FF;

}

.cards:hover{

box-shadow: 1px 8px 20px grey; -webkit-transition: box-shadow .2s ease-in;

18

}

/\* document css \*/

.single-chart {

width: 33%;

justify-content: space-around ; }

.circular-chart {

display: block;

margin: 10px auto;

max-width: 100%;

max-height: 250px;

}

.circle-bg {

fill: none;

stroke: #eee;

stroke-width: 3.8;

}

.circle {

fill: none;

stroke-width: 2.8;

stroke-linecap: round;

animation: progress 1s ease-out forwards; }

19

@keyframes progress { 0% { stroke-dasharray: 0 100; }

}

.circular-chart.orange .circle { stroke: #ff9f00;

}

.circular-chart.green .circle { stroke: #4CC790;

}

.circular-chart.blue .circle { stroke: #3c9ee5;

}

.percentage {

fill: rgb(255, 255, 255);

font-family: sans-serif;

font-size: 0.5em;

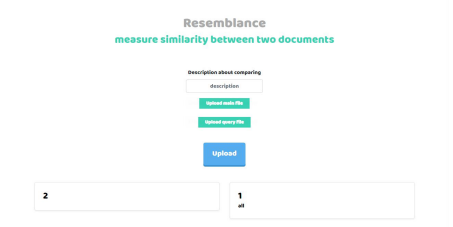
text-anchor: middle;

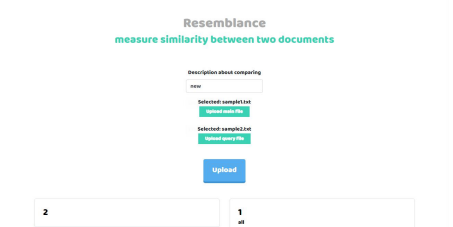
}

20

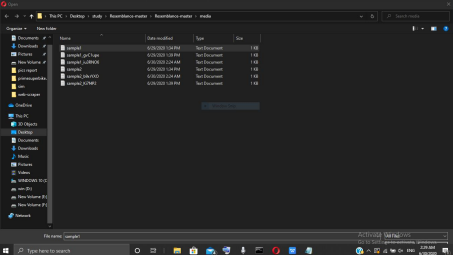
**CHAPTER - 5**

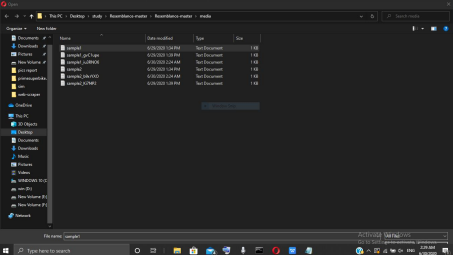
**OUTPUT**

****Fig 5.1 Main Page

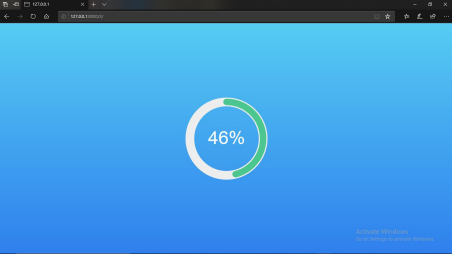
Fig 5.2 Data Entry

21

Fig 5.3 Uploading File 1

Fig 5.4.Uploading File 2

22

Fig. 5.5 Result Display in Percentage using GUI

23

**CHAPTER - 6**

**APPLICATIONS**

Application of text similarity is to comapre data and check for data duplicationinasystem and improve defragmentation of extra memory usage. It also plays animportantrole for E-Commerce and E-Service sectors to understand their web sites andserviceareused and provide better service for both customers and users.

1] Reducing Duplication in data

2] E-Verification of data

3] E –Government

4] Electronic commerce

5] E-Politics and E-Democracy

6] Security and Crime Investigation

7] Electronic Business

24

**CHAPTER - 7**

**CONCLUSION**

We have successfully demonstrated how to extract data form text files and compareitscontents with each other to find similarity in two different documents. In this project wehavedisplayed the different ways to tokenize the text into tokens and sort it accordingtoitsnumerical value and find numerical distance between the contents by its assignednumericalvalue. We have used a functions for taking the average of similarity between the contentsoffile saved in different dictionaries in Python and displaying the result using GUI

25

**CHAPTER - 8**

**FUTURE SCOPE**

Research activities on this topic have drawn heavily on techniques developedinotherdisciplines such as Research activities on this topic have drawn heavily ontechniquesdeveloped in other disciplines such as Information Retrieval (IR) and Natural LanguageProcessing(NLP). There exists significant body of work in extracting knowledge fromfilesinthe field of Text Processing and Analysis,the application of these techniques to webcontentmining has been limited .

26

**CHAPTER - 9**

**REFERENCES**

[1] https://lil.law.harvard.edu/blog/2019/10/22/guest-post-creating-a-case-recommendation-system-using-gensims-doc2vec/.

[2] https://www.guru99.com/tokenize-words-sentences-nltk.html”.

[3] https://www.oreilly.com/library/view/applied-text-analysis/9781491963036/ch04.html

[4] Steven Bird, Ewan Klein, and Edward Loper “Natural Language Processing withPython– Analyzing Text with the Natural Language Toolkit”

27