**BDAD Project**

**Report**

**On**

**Advanced Speech Data Processing**

### Developed By: -

Het Mehta(18162121008)

Aditya Lakkad(18162121010)

### Guided By:-

Prof. Sulabh Bhatt

**Submitted to**

**Department of Computer Science & Engineering Institute of Computer Technology**

****

**Year: 2021**



# CERTIFICATE

This is to certify that the Big Data Application Development Project work entitled **“MAD-I : Voice Based advertisement solution”** by Het Mehta(Enrolment No.18162121008), Aditya Lakkad(Enrolment No.18162121008) of Ganpat University, towards the fulfillment of requirements of the degree of Bachelor of Technology – Computer Science and Engineering, carried out by them in the CSE(BDA) Department. The results/findings contained in this Project have not been submitted in part or full to any other University / Institute for award of any other Degree/Diploma.

Name & Signature of Internal Guide

Name & Signature of Head

**Place: ICT - GUNI**

### Date:

**ACKNOWLEDGEMENT**

BDAD project is a golden opportunity for learning and self-development. I consider myself very lucky and honored to have so many wonderful people lead me through in completion of this project. First and foremost, I would like to thank **Dr. Hemal Shah,** Head of Department, Computer Science and Engineering, who gave us an opportunity to undertake this project. My grateful thanks to **Prof. Sulabh Bhatt** for their guidance in project work **MAD-I : Voice Based advertisement solution**, who despite being extraordinarily busy with academics, took time out to hear, guide and keep us on the correct path. We do not know where would have been without his help. CSE department monitored our progress and arranged all facilities to make life easier. We choose this moment to acknowledge their contribution gratefully.

**Het Mehta (Enrollment No:182162121008)**

**Aditya Lakkad (Enrollment No.:18162121010)**

**ABSTRACT**

One application of Big Data Analytics has been suggestion of appropriate advertisements as per data received. For example, if someone searches for shoes, there should be advertisements for shoes and its related products such as socks or shoe shine, etc. This application intends to gather data and analyse and extract key information and then show us appropriate advertisements.

**INDEX**

### Title PageNo

**CHAPTER 2:PROJECT SCOPE 03-04**

**CHAPTER1:INTRODUCTION 01-02**

**CHAPTER 3: SOFTWARE ANDHARDWAREREQUIREMENT 05-06**

**CHAPTER 4:PROCESS MODEL 07-08**

[CHAPTER 5: PROJECT PLAN 09-10](#_TOC_250009)

* 1. [List of MajorActivities 10](#_TOC_250008)
  2. [Estimated Time Durationin Days 10](#_TOC_250007)

[CHAPTER 6: IMPLEMENTATION DETAILS 11-](#_TOC_250006)21

* 1. [Database Model. 12](#_TOC_250005)
  2. Development of the login. 12
  3. Development of the index page. 14
  4. Development of NLP Pipeline. 17
  5. Development of ETL Pipeline. 20

**CHAPTER 7: CONCLUSION AND FUTURE WORK 22-23**

**CHAPTER8:REFERENCE 24-25**

**CHAPTER: 1 INTRODUCTION**

**CHAPTER 1 INTRODUCTION**

Data science has many applications in the real-life world, one of them being in marketing sector. Marketing sector consists of many subsections, but the one we will be focusing on is advertisements, in particular advertisement suggestions from a speech recorded. Such system already exists in applications such as Facebook or Instagram. They extract data from your google searches or the location you are at or cross-referencing the products bought from many people in contact with reference of one person. These types of applications require extensive processing power and a very large database along with permissions to access data from other applications. In the application we developed the scenario is such that if two people are talking among themselves, the application will record the data and suggest advertisements based on the keywords extracted from the speech. The advertisements are shown from an online database along with images. The advertisements are shown on a web application developed by us.

**CHAPTER: 2 PROJECT SCOPE**

### CHAPTER 2 PROJECT SCOPE

Currently, the application is limited to the web application we developed but it can be extended to other applications in future.

# CHAPTER: 3 SOFTWARE AND HARDWARE REQUIREMENTS

### CHAPTER 3 SOFTWARE AND HARDWARE REQUIREMENTS

**Minimum Hardware Requirements**

|  |  |
| --- | --- |
| **Microphone** | Required |

*Table 3.1 Minimum Hardware Requirements*

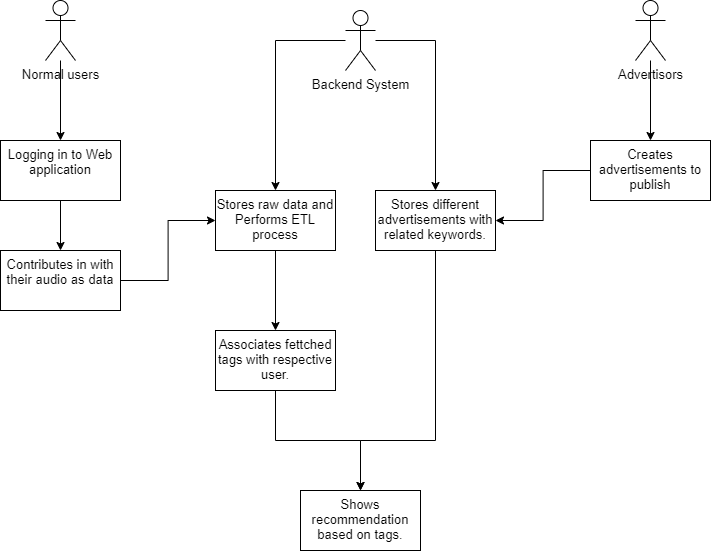
**Minimum Software Requirements**

|  |  |
| --- | --- |
| **Operating System** | Any operating system which can support an internet browser. |
| **Other tools & tech** | Internet browser |

*Table 3.2 Minimum Software Requirements*

**CHAPTER: 4 PROCESS MODEL**

### CHAPTER 4 PROCESS MODEL

****

*Figure 4.1 Process Model of Project*

## CHAPTER: 5 PROJECT PLAN

### CHAPTER 5 PROJECT PLAN

### List of Major Activities

### Frontend task: Done by Aditya Lakkad

Task: - 1 Designing and coding html frontend.

Task: - 2 Research for speech recognition.

Task: - 3 Coding and Testing JavaScript Speech recognition.

Task: - 4 Creating Flask and Mongodb backend.

Task: - 5 Implementing an API for pushing voice data into database.

Task:- 6 Implementing a Pipeline to perform ETL with het’s prepared NLP code.

### Backend task: Done by Het Mehta

Task: - 1 Initial speech to text process in python

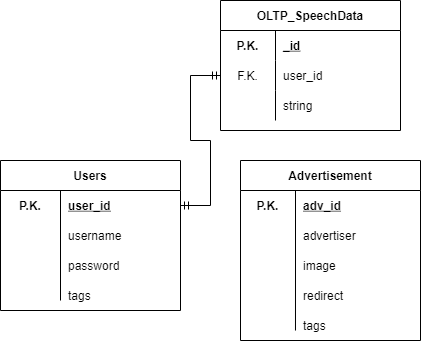
Task: - 2 Research and implementation of TF-IDF for English language

Task: - 3 Refining the output

# CHAPTER: 6 IMPLEMENTATION DETAILS

### CHAPTER 6 IMPLEMENTATION DETAIL

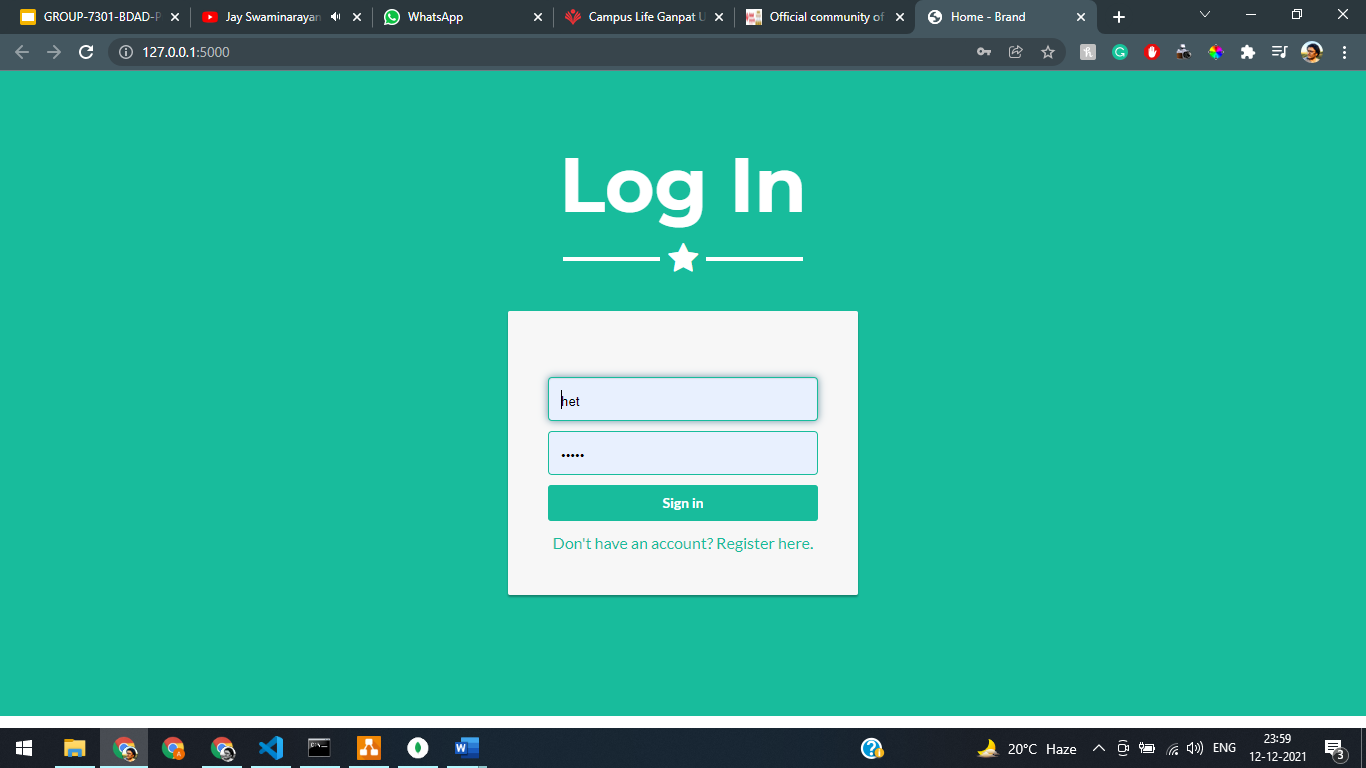
### Database model

****

*Figure 6.1 Data Model*

* 1. **Development of the Login page.**

Here, We have used HTML5 and CSS3 with Bootstrap5 framework to design good looking and responsive webpages. Here for development of this frontend of the web application. The first and the default page was of the login page.



User can use their provided username and password to login to the web application. This details has been stored in the users table. In the backend, using flask framework and python we are verifying a valid and authenticate user login.

@app.route('/login', methods=['POST'])

def login():

*if* request.method == 'POST':

        users = mongo.db.users

        login\_user = users.find\_one({'username' : request.form['username'], 'password' : request.form['password']})

        ads = mongo.db.advertisements.find({})

        tags = login\_user['tags']

        print(login\_user)

After verifying a successful user login, we will implement a best search algorithm and based on that we will find a best fit advertisement for that particular user and get’s total of that three tags and passes it to the further route of index.

*if* login\_user:

            x = True

            l = []

            link = []

*if* len(tags < 3):

                link = ["www.adityalakkad.ml","www.adityalakkad.ml","www.adityalakkad.ml"]

                l = ["cake.png","cake.png","cake.png"]

*return* render\_template('index.html',img=list(l),link=list(link),lu=login\_user['\_id'])

*while* x:

*for* i *in* ads:

*for* j *in* tags:

*if* j in i['tags']:

*if* i['image'] not in l:

                                l.append(i['image'])

                                link.append(i['redirect'])

*if* len(l) >= 3:

                            x = False

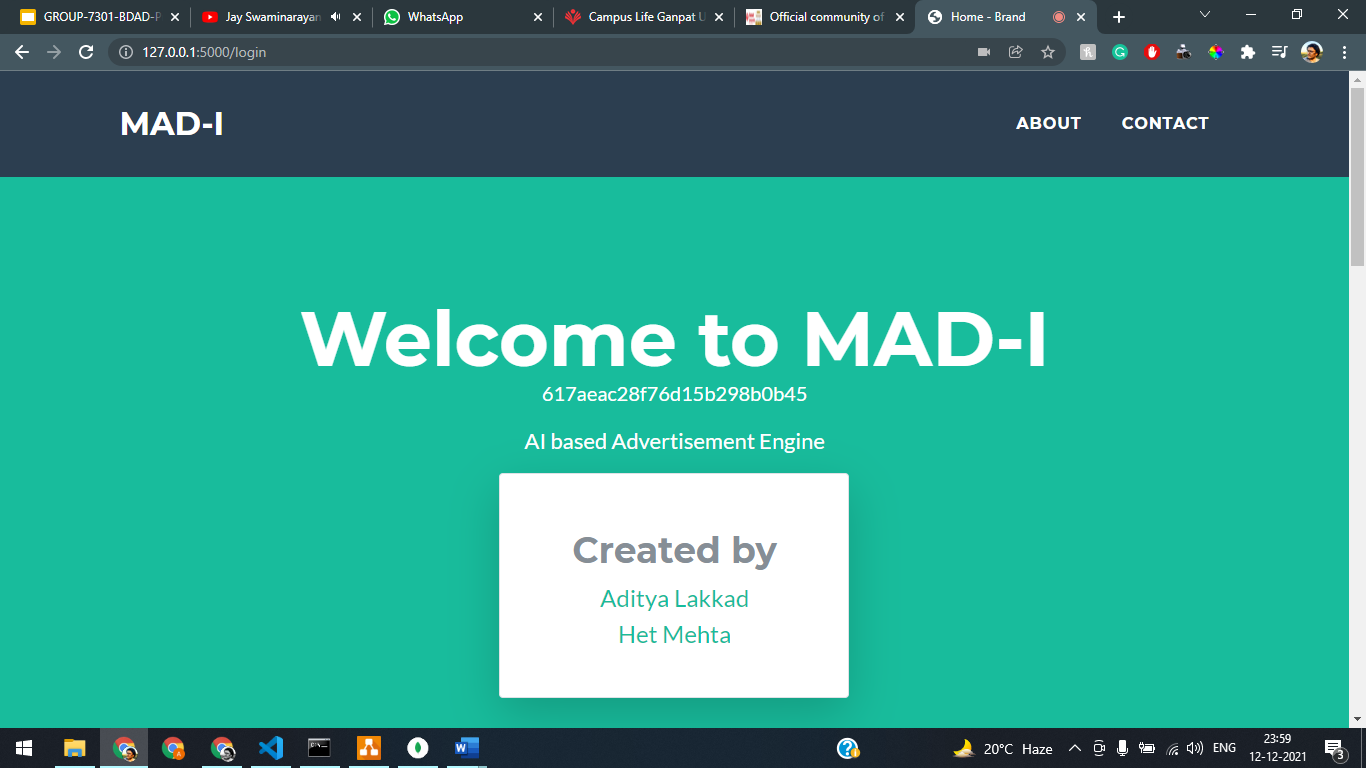
*break*

*return* render\_template('index.html',img=list(l),link=list(link),lu=login\_user['\_id'],cake="cake.png")

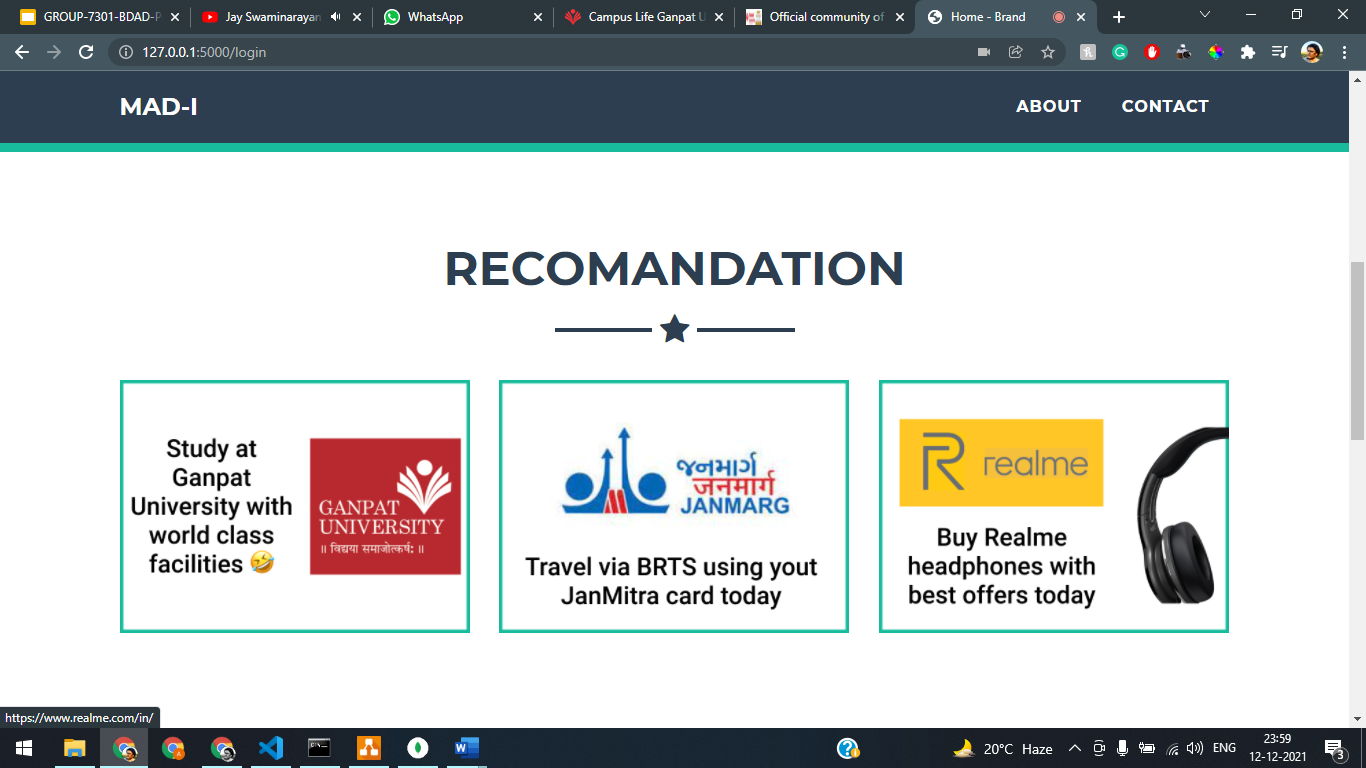
As we can see here at max total of 3 best fit advertisement recommendation will be passed to the further. And the audio data of a user also will be provided on the index page.

* 1. **Development of the Index page.**

The index page is the primary page of the web application here lies the code of fetching the audio data of user and sending it to the backend to store it in the OLTP\_SpeechData database. Here also we shows some recommendation that has been provided us by the login page so let’s first see about index page. Also built using BootStrap5 and HTML5, basic JavaScript for animations.



From the login route we have been provided with three advertisement recommendations.



This recommendation has been generated based on the best fit of the advertisement tags and tags associated with user. These associated tags are based on the user’s speech data.

Now to fetch this audio data we need some mechanism to extract data from frontend convert that into speech data and some how send it to the backend. To extract speech data and convert into the text we are using basic JavaScript modules and their inbuilt functions to achieve our goals.

Here is the code for that which is written in JavaScript and implemented into index.html file.

const texts = document.querySelector(".texts");

window.SpeechRecognition =

  window.SpeechRecognition || window.webkitSpeechRecognition;

const recognition = new SpeechRecognition();

recognition.interimResults = true;

First of all we are using speech recognition module and it’s SpeechRecognition function which we will use to recognize user’s speech from the raw audio data.

recognition.addEventListener("result", (e) => {

  const text = Array.from(e.results)

    .map((result) => result[0])

    .map((result) => result.transcript)

    .join("");

  var logedinUserId = $('#my-data').data();

  console.log(logedinUserId);

*if*(e.results[0].isFinal){

    text.textContent = text;

    console.log(text);

    const xhr = new XMLHttpRequest();

    xhr.open('POST', 'http://127.0.0.1:5000/api/v1/pushString');

    xhr.setRequestHeader('Content-Type', 'application/json');

    xhr.onload = () => {

      console.log('Sent');

    };

    console.log('{{lu}}');

    xhr.send(JSON.stringify({

      'user\_id': '{{lu}}',

      'string': text

    }));

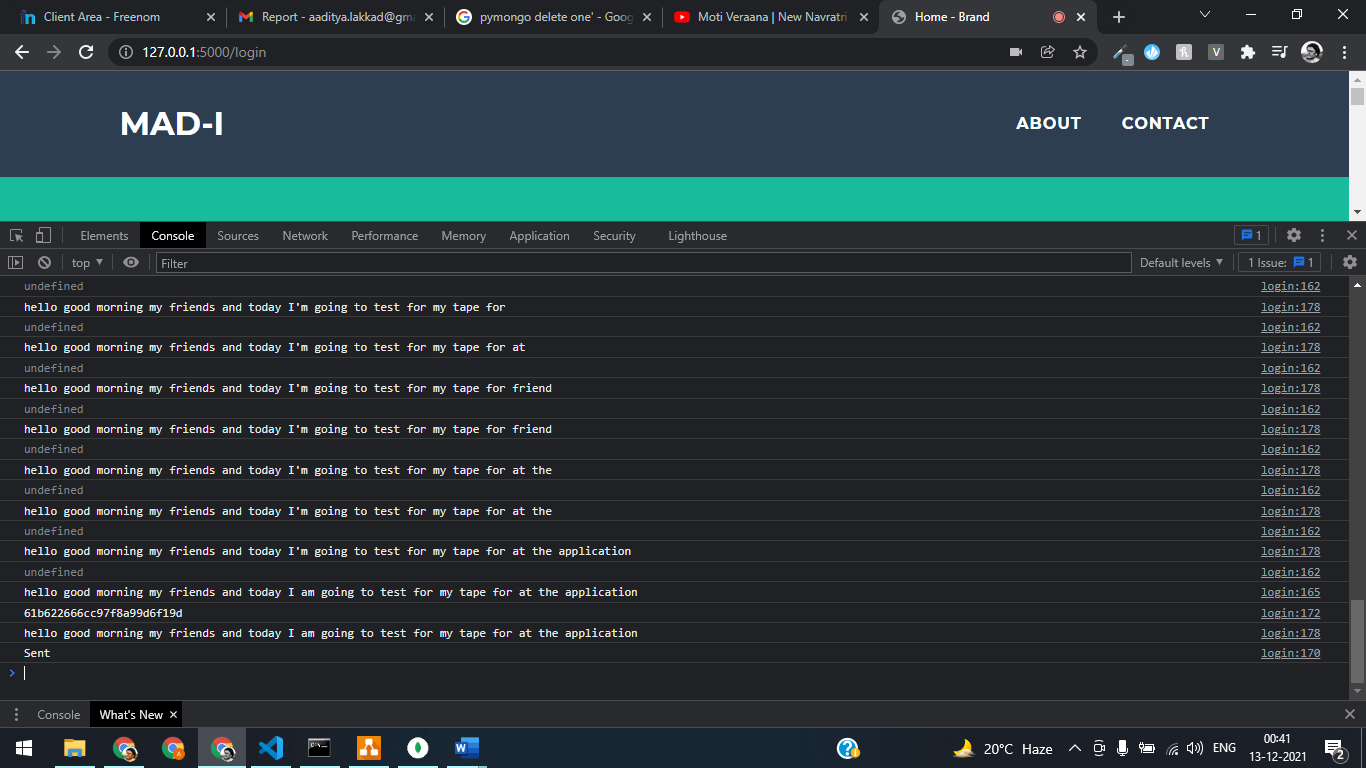
  }

  console.log(text);

});

After that we are recognizing the language and converting audio into text and sending that text via an API to the backend.

Now let’s take a glimpse at the output that how it is looking in action



Here we can see that my voice has been used to generate text with the accuracy of 97% and that text has been sent to the backend using our pushString api that has been created with the flask application. So let’s look at that api.

@app.route('/api/v1/pushString', methods=['POST'])

def pushString():

    print(request.json)

    table = mongo.db.OLTP\_SpeechData

    s = table.find\_one({'user\_id':request.json['user\_id']})

*if* s:

        temp = s['string']

        table.update({'user\_id':request.json['user\_id']},{'$set':{'string':temp + request.json['string']}})

*else*:

        userid = request.json['user\_id']

        string = request.json['string']

        table.insert({'user\_id':userid, 'string':string})

*return* jsonify({'result':'success'})

This API interacts with the OLTP\_SpeechData table to add text string of the user. Here, if a particular user is already in the table then it will append within the same user’s string. And if there is no same user it will simply create a new entry.

* 1. **Development of the NLP Pipeline.**

TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.

TF-IDF = TF(t,d) \* IDF(t)

Terminology –

t – term/word

d – document

N - Number of documents

corpus – the set of documents

Since we are only taking one continuous speech number of documents will be 1.

def computeTF(wordDict,bow):

*'''Computing TF(Term Frequency of the vocab) '''*

*try*:

        tfDict = {}

        bowCount = len(bow)

*for* word, count *in* wordDict.items():

            tfDict[word] = count/float(bowCount)

*return* tfDict

*except* ZeroDivisionError:

        print('No recognisable nouns detected please try again.')

*return* False

TF(t,d) = count of t in d/ number of words in d

In simple words, we are getting the ratio of count of a particular word upon the total number of words. For example in the sentence – “Data science is the analysis of data to get insights“,

TF(‘data’) = 2/10

def computeIDF(doclist):

*'''Computing IDF for the vocab '''*

*import* math

    count = 0

    idfDict = {}

*for* element *in* doclist:

*for* j *in* element:

            count+=1

    N = count

    idfDict = dict.fromkeys(doclist[0].keys(),0)

*for* doc *in* doclist:

*for* word,val *in* doc.items():

*if* val>0:

                idfDict[word]+= 1

*for* word,val *in* idfDict.items():

*if* val == 0:

            idfDict[word] = 0.0

*else*:

            idfDict[word] = math.log(N / float(val))

*return* idfDict

IDF(t) = log(N/ df(t))

Here, df(t) = occurrence of t in documents

In simple words, it measures informativeness of term t, IDF is will be very low for common words such as stop words is,the,A,etc. Hence, they will be unimportant for overall data extraction allowing us to focus more on topic words.

def computeTfidf(tf,idf):

    tfidf = {}

    sorted\_list = []

*for* word , val *in* tf.items():

        tfidf[word] = val \* idf[word]

    ranking\_list  = sorted(tfidf.items(),reverse=True, key = lambda kv:(kv[1], kv[0]))[:10]

*for* i, \_ *in* ranking\_list:

        sorted\_list.append(i)

*return* sorted\_list

def topic\_ext(text):

    vocab\_dict , arr = textProcessing(text)

    tf = computeTF(vocab\_dict,arr)

*if*(tf != False):

        idf = computeIDF([vocab\_dict])

        tfidf = computeTfidf(tf,idf)

*return* tfidf

We have to get key words from a single sentence; hence we have to do extra filtering to get accurate output. For this, we have applied filter that will only return nouns and proper nouns. We accomplish that through POS tagging. We convert all words from the sentence to upper case so as to avoid problems with upper- and lower-case words, next we add all words that are classified as nouns or proper nouns and also their child nodes if any, to a list. Later if there are any duplicates, we remove those as well.

def textProcessing(doc):

    Nouns = []

    Noun\_set = []

    trimmed\_noun\_set = []

    removing\_duplicates = []

    arr = []

    vocab = []

    vocab\_dict = {}

    doc = nlp(doc.upper())

*for* possible\_nouns *in* doc:

*if* possible\_nouns.pos\_ in ["NOUN","PROPN"] :

            Nouns.append([possible\_nouns , [child *for* child *in* possible\_nouns.children]])

*for* i,j *in* Nouns:

*for* k *in* j:

            Noun\_set.append([k,i])

*for* i , j *in* Noun\_set:

*if* i.pos\_ in ['PROPN','NOUN','ADJ']:

            trimmed\_noun\_set.append([i ,j])

*for* word *in* trimmed\_noun\_set:

*if* word not in removing\_duplicates:

            removing\_duplicates.append(word)

*for* i *in* removing\_duplicates:

        strs = ''

*for* j *in* i:

            strs += str(j)+" "

        arr.append(strs.strip())

*for* word *in* Noun\_set:

        string = ''

*for* j *in* word:

            string+= str(j)+ " "

        vocab.append(string.strip())

*for* word *in* vocab:

        vocab\_dict[word]= 0

*for* word *in* arr:

        vocab\_dict[word]+= 1

*return* vocab\_dict , arr

The output we got from the above function is passed on to TF function, we get the ratio as output. If in an unusual case, there are no nouns or proper nouns identified in the sentence it will give us an ‘No recognisable nouns detected please try again.’ Message and not carry out the remaining function

After that we calculate the output of IDF function, and save it. And finally, we multiply both outputs to get our TF-IDF output. An additional step is we sort the output value for each term in descending way so it gives us the most relevant term first and least relevant term last, and then match those values with their terms to give us the aforementioned terms in descending order of their TF-IDF values. And this is how we get keyword extraction from a sentence

* 1. **Development of the ETL Pipeline.**

Now we are developing an ETL pipeline, which can extract data from the OLTP\_SpeechData data table and transform it into tags using our NLP Pipeline , and at last we will load that transformed data into the our main users data table. Let’s see how we have created the pipeline

*from* pymongo *import* MongoClient

*from* bson.objectid *import* ObjectId

client = MongoClient('mongodb://localhost:27017/MAD\_I\_BDAD')

table = client['MAD\_I\_BDAD']['OLTP\_SpeechData']

table2 = client['MAD\_I\_BDAD']['users']

We have imported the necessary modules. Using pymongo module I have connected with the mongodb instance running on my localhost and got necessary table from our mongo db client object.

def extract():

    data = table.find()

*return* data

Here is our extract process which is nothing but a simple function call to get all the data from our OLTP\_SpeechData data table. In future it can be further modified more advanced filtering and extraction option.

def transform(data):

    transformed\_data = {}

*for* i *in* data:

        transformed\_data[i['user\_id']] = topic\_ext(i['string'])

        deli = table.delete\_one({'user\_id': i['user\_id']})

        print(deli)

*return* transformed\_data

Transform process will be using the string data along with the user id, it extracts keywords from the text using NLP Pipeline function which will return a list of the keywords. Which will be passed on to the Load function. There finally from Raw text data meaningful keywords will be loaded into our data table of users.

def load(t):

*for* i *in* t:

        print(i)

        temp = table2.find\_one(ObjectId(i))

*if* len(temp['tags']) >= 100:

            s = table2.update\_many(

                { "\_id": ObjectId(i) },

                { "$push": { "tags": { "$each": [], "$slice": len(temp['tags']) } }

             }

            )

*for* j *in* t[i]:

                table2.update\_many(

                    { "\_id": ObjectId(i) },

                    {'$push': {'tags': j}}

                )

*else*:

*for* j *in* t[i]:

                table2.update(

                    { "\_id": ObjectId(i) },

                    {'$push': {'tags': j}}

                )

*return* 0

Here, in the load process we are getting the data from transform process which contains meaningful tags(keywords) extracted from the raw text data and their appropriate user id after that we will find that particular user and from there we will check if there is greater than 100 tags if yes then ww will slice the array and add up our extracted keywords in the array.

**CHAPTER: 7 CONCLUSION AND FUTURE WORK**

### CHAPTER 7 CONCLUSION AND FUTURE WORK

**Conclusion**

To reiterate my views, In the sale of any kind of product a personal touch always add ups a positive point in the process. With our application we are trying to provide just that mechanism that we can use the audio data to recommend the users their personalized ads, And with that we can increase our customer relationship values. Using our application we can also analyze the customer needs and in the future we can make our recommendation strong which will help to increase our sale.

### Future work

Currently we have a small scale of data that’s why the simulation comes up seamlessly, with the increase in the users we will have lot more data to deal with and we have to implement more optimized algorithm to find recommendations and optimized data engineering pipelines that we can scale according to our need

* Scalable Data pipelines
* Better speech recommendation
* Multi language support
* Optimized algorithm for recommendations.

# CHAPTER: 8 REFERENCES

### CHAPTER 8 REFERENCES

1. <https://flask.palletsprojects.com/en/2.0.x/>
2. <https://pymongo.readthedocs.io/en/stable/>
3. <https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition>
4. <https://www.mongodb.com/developer/how-to/flask-python-mongodb/>