1. INTRODUCTION

1.1. OVERVIEW

CBIT DISCUSSION FORUM is a platform where every student can come up with their issues in the college with respective suggestions and we make sure that their issues are reported and a solution to be found by making the issue reach the faculty.

1.2. AIM OF THE PROJECT

The main aim of our project is to bring the entire college to a common platform where they can discuss their issues openly and anaoumously so that their fear of complaining the issues to the faculty and also the tedious process of writing a letter ad going from one section of office to other wont be there. This project aims to bring out our college issues ad be resolved to both college and students betterment.

1.3. ORGANISATION OF REPORT

The organization of the report is as follows:

Chapter 1 deals with the Introduction of the project and gives the details about the project in an abstract view.
Chapter 2 deals with the information about NLP ad DJANGO framework and its utilization details are discussed in brief.
Chapter 3 deals with the Software Requirements Specifications which is a specification of the project software and hardware requirements.
Chapter 4 deals with the Implementation part which includes the tools and software that are used.
Chapter 5 deals with the Testing of the project and screenshots of the project
Chapter 6 explains the Conclusion and further scope of the project.

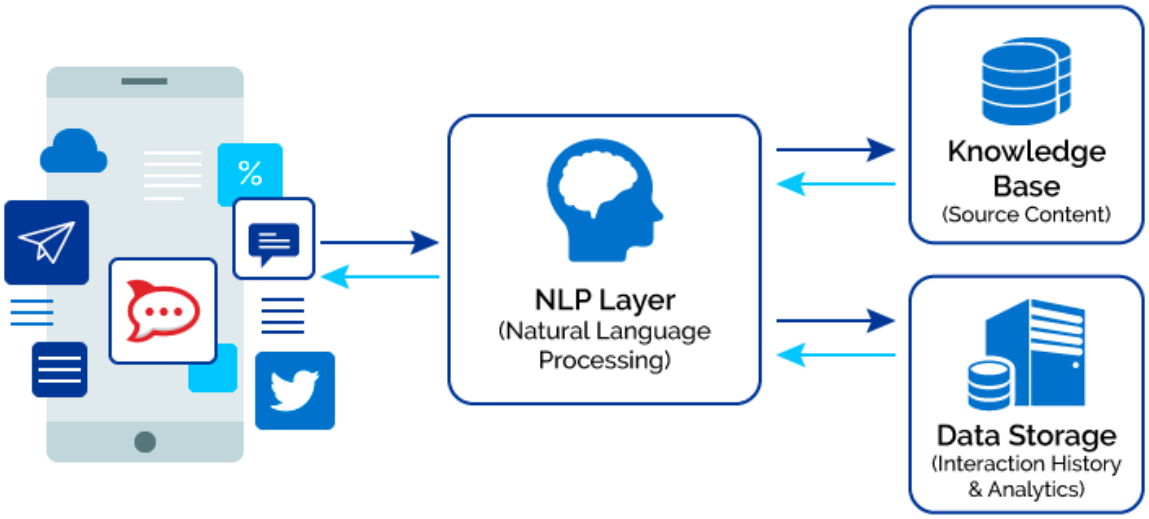
2. TECHNOLOGIES

2.1.NLP

Natural language processing (NLP) is the ability of a computer program to understand human language as it is spoken. NLP is a component of artificial intelligence (AI).

The development of NLP applications is challenging because computers traditionally require humans to "speak" to them in a programming language that is precise, unambiguous and highly structured, or through a limited number of clearly enunciated voice commands. Human speech, however, is not always precise -- it is often ambiguous and the linguistic structure can depend on many complex variables, including slang, regional dialects and social context.

NLP involves communication through speech and text. Speech communication is the recent innovation that includes chatbots and other voice-based bots. These are designed as human language personal assistant. You may be already familiar with the personal assistant found on iPhone's Siri – a personal assistant that just communicates like a human and can assign her almost any task you wish to do. Instructions like calling a friend, find restaurants and events, check weather and so on. The list is endless. It may even tweet and update your status on facebook just like a human friend with incredible intelligence.



2.1. figure depicting steps involved in NLP

NLP has strengthened the interactions with the search engines that allowed the queries to be assessed faster and in an efficient manner. The most important part of it is, it understands the query given to it and fetches accurate results.

NLP has given rise to voice search that becomes increasingly popular these days. And Google study of 2014 reveals 55% of teens and 41% of adults in the US use voice search more than once a day. In 2016, Google announced that 20 percent of its mobile app and Android devices are voice searches. All these numbers are bound to increase drastically in the coming

years. It is the increasing speed of computer processing that made the voice search a possibility and now an increasing popularity.

2.1.1 IMPORTANCE OF NLP

The development of NLP applications is challenging because computers traditionally work with precise, unambiguous and highly structured languages such as programming languages, however, the natural language is often ambiguous and the linguistic structure can depend on many complex variables, including slang, regional dialects and social context.

With the ongoing growth of the World Wide Web and social media, there is a drastic increase in online data. As the amount of data increases the mechanisms to process these unstructured data and to extract meaningful information from it becomes more challenging. It will be very difficult to find a specific piece of information from a large knowledge base. These challenges and difficulties can be overcome with the advanced NLP techniques.

The most widely used NLP application is machine translation which helps to overcome the language barriers. As the amount of information available online is increasing day by day, the need to access and process it becomes more important. To convert information from one language to another, machine translation can be used. The NLP techniques help the machine to understand the meaning of sentences, which improves the efficiency of machine translation.

The NLP techniques are very useful for sentiment analysis. It helps to identify the sentiment among several online posts and comments. The business firms make use of NLP techniques to know about the customer's opinion about their product and services from the online reviews.

Using NLP the Automatic summarization can be performed more efficiently. Automatic summarization is relevant not only for summarizing the meaning of documents and information but also for understanding the emotional meanings of the information, such as in collecting data from social media. Automatic summarization is especially relevant when used to provide an overview of a news item or blog posts while avoiding redundancy from multiple sources and maximizing the diversity of content obtained. Using this, the difficulty to find an important piece of information from a huge knowledge base can be reduced.

The advanced NLP techniques allow the non-programmers to interact with the computing systems and obtain useful information from it. Using NLP the common synonyms for the input phrases can be detected and match them with the right answers, it helps the users who are unfamiliar with the terminologies of the computing system. Spam filtering, language understanding, text classification, information extraction, question answering, social website feeds, Voice recognition and speech-to-text are the other typical applications of NLP.

There are many open source Natural Language Processing (NLP) libraries and these are some of them:

- Natural language toolkit (NLTK)
- Gate NLP library
- Apache OpenNLP.
- Stanford NLP suite
- MALLET

NLTK is more popular and the leading platform for building NLP applications, which is written in python. It provides an intuitive framework along with substantial building blocks, consistent interfaces and data structures.

2.1.2. Different ways NLP can help:

Text Recognition

Recognizes the text, character and converts them into data and stores it in its database. The ability to read the text and converts into a machine-encoded text is one of the methods of language processing that is been used by search engines for many years. With the help of NLP, things are far easier than it was before in terms speed and accuracy. No more mere search results but it give you the answers to the question posed by the customers.

Semantics

We, humans, understand the word in the context of the sentence spoken or written and we do it more efficiently and effortlessly. But to teach the computer the context in which the sentence is spoken or written is quite a task. Machines do not understand what and the why.

As we all know that training makes us perfect in something we do, the same theory applies here as well to the computer world. They have been given a lot of unstructured data for semantic analysis and through powerful algorithms; computers are becoming more powerful and getting better at understanding the particular word in reference to the context or scenario to comprehend the phrase.

Sentiment Analysis

When do you know what exactly means ‘happy customer experience’ is, which is very much subjective. Even, if we find out the ways to get into it, how to teach the systems to understand the emotions of the text? Yes, things are still in the primitive stage to evaluating the customer views that may be made available to us through our research team. Customer feedbacks, answers to queries, their likes, and dislikes, their choice and preferences in the coming festival seasons, holidaying trends, better product ideas, their expectations with regard to the product and services, etc., will amount to a huge unstructured data.

To analyze the enormous amount of unstructured data and interpret the outcome of such reviews requires huge manpower. But with computers now tuned to AI, customer’s emotional responses,

analysis, and findings are marked as a positive, negative or neutral outcome. It would be easier for the computers to understand the simple answers or interactions. However, complex responses complicate the whole comprehension of machine learning. But there are several methods to segregate the complicated words from complex sentence patterns to determine the exact meaning of the sentences. Thus, this further provides a high level of accuracy in predicting the ambiguous phrases in simpler ways.

Personalized Bots

An organization may reap maximum benefits using NLP in designing personal assistants. Shopping can be more fun than ever. These assistants have the ability to keep the customer interested and bring on their screen exactly what they require to shop. It analyses recent searches you made, past purchase behavior to bring out seamless shopping experience.

NLP would be able to make machine talking to a human in the easiest possible way. Chatbots technology may be used in business houses to extract information from past data that might help taking big business decisions. The powerful technology also helps you forecast the revenues based on the statistical data in a flash. The insights delivered by the chatbots may transform your business into a formidable platform to find the right answers to leap into the future.

2.1.3 How natural language processing works

Current approaches to NLP are based on deep learning, a type of AI that examines and uses patterns in data to improve a program's understanding. Deep learning models require massive amounts of labeled data to train on and identify relevant correlations, and assembling this kind of big data set is one of the main hurdles to NLP currently.

Earlier approaches to NLP involved a more rules-based approach, where simpler machine learning algorithms were told what words and phrases to look for in text and given specific responses when those phrases appeared. But deep learning is a more flexible, intuitive approach in which algorithms learn to identify speakers' intent from many examples, almost like how a child would learn human language.

2.2 NATURAL LANGUAGE TOOL KIT

NLTK is a leading platform for building Python programs to work with human language data. 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 NLP libraries, and an active discussion forum.

NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the

fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more.

2.2.1 Getting Started with NLTK

In our project we are striving to use nltk to preprocess our data and mould it to required format so that we can apply different algorithms to arrive at our output

First we need to download

```
import nltk

nltk.download()
```

2.2.2 Modules used in nltk

Profanity :

A Python library to check for (and clean) profanity in strings. Load your own wordlist, or use the bundled one. Censors words using standard censorcharacters (!@#$%), or load your own censor Uses a pool of censor characters to create a more natural censor string. Censors all instances of a given word with the same censor string - for easy correlation.

Stop Words:

Text may contain stop words like ‘the’, ‘is’, ‘are’. Stop words can be filtered from the text to be processed. There is no universal list of stop words in nlpresearch, however the nltk module contains a list of stop words.

```
from nltk.corpus import stopwords
```

We get a set of English stop words using the line:

```
stopWords=set(stopwords.words('english'))
```

Tokenize:

Tokenize words:A sentence or data can be split into words using the method word_tokenize():

All words except the comma. Special characters are treated as separate tokens.

Tokenizing sentences:The same principle can be applied to sentences. Simply change the to sent_tokenize()

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

NLTK speech tagging:

The module NLTK can automatically tag speech. Given a sentence or paragraph, it can label words such as verbs, nouns and so on.

```
import nltk

from nltk.tokenize  import  PunktSentenceTokenizer
```

ngrams :

Bigrams are 2-contiguous word sequences. Trigrams are 3-contiguous words.We can uses nltk.collocations.ngrams to create ngrams. Depending on the n parameter, we can get bigram, trigram, or any ngram. The function returns a generator object and it is possible so create a list, for example A = list(A).

```
    from nltk.util import ngrams
```

LDA(Latent Dirichlet Allocation):

To generate an LDA model, we need to understand how frequently each term occurs within each document. To do that, we need to construct a document-term matrix with a package called gensim:

```
from gensim import corpora, models

dictionary = corpora.Dictionary(texts)
```

The Dictionary() function traverses texts, assigning a unique integer id to each unique token while also collecting word counts and relevant statistics. To see each token's unique integer id, try print(dictionary.token2id).

Next, our dictionary must be converted into a bag-of-words:

```
corpus = [dictionary.doc2bow(text) for text in texts]
```

The doc2bow() function converts dictionary into a bag-of-words. The result, corpus, is a list of vectors equal to the number of documents. In each document vector is a series of tuples.

corpus is a document-term matrix and now we're ready to generate an LDA model:

```
ldamodel = gensim.models.ldamodel.LdaModel(corpus, num_topics=3, id2word = dictionary, passes=20)
```

The LdaModel class is described in detail in the gensim documentation. Parameters used :

Parameters:

- num_topics: required. An LDA model requires the user to determine how many topics should be generated. Our document set is small, so we're only asking for three topics.
- id2word: required. The LdaModel class requires our previous dictionary to map ids to strings.
- passes: optional. The number of laps the model will take through corpus. The greater the number of passes, the more accurate the model will be. A lot of passes can be slow on a very large corpus.

What does lda do?

Once you provide the algorithm with the number of topics, all it does it to rearrange the topics distribution within the documents and keywords distribution within the topics to obtain a good composition of topic-keywords distribution.A topic is nothing but a collection of dominant keywords that are typical representatives. Just by looking at the keywords, you can identify what the topic is all about.

The following are key factors to obtaining good segregation topics:

1. The quality of text processing.
2. The variety of topics the text talks about.
3. The choice of topic modeling algorithm.
4. The number of topics fed to the algorithm.
5. The algorithms tuning parameters.

2.3 DJANGO FRAMEWORK

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. Because Django was developed in a fast-paced newsroom environment, it was designed to make common Web-development tasks fast and easy.

The layers and model methods used related to Django are:

a) The model layer: Django provides an abstraction layer (the "models") for structuring and manipulating the data of your Web application. A model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you're storing.

all(): The simplest way to retrieve objects from a table is to get all of them. To do this, use the all() method on a Manager: This method returns a QuerySet of all the objects in the database.

b) Fields: Each field in your model should be an instance of the appropriate Field class. Django uses the field class types to determine the column type and minimum validation requirements.

c)Views: A view function, or view for short, is simply a Python function that takes a Web request and returns a Web response.For the sake of putting the code somewhere, the convention is to put views in a file called views.py , placed in your project or application directory

GET and POST: GET and POST are the only HTTP methods to use when dealing with forms. Django's login form is returned using the POST method, in which the browser bundles up the form data, encodes it for transmission, sends it to the server, and then receives back its response. GET, by contrast, bundles the submitted data into a string, and uses this to compose a URL. The URL contains the address where the data must be sent, as well as the data keys and values.

render(): Combines a given template with a given context dictionary and returns an HttpResponse object with that rendered text.

render(request, template_name, context=None, content_type=None, status=None, using=None)

d)URLconf: A URLconf is like a table of contents for your Django-powered web site. Basically, it's a mapping between URLs and the view functions that should be called for those URLs. It's how you tell Django, "For this URL, call this code, and for that URL, call that code."

url(*regex, view, kwargs=None, name=None*)

e) Migrations: Migrations are Django's way of propagating changes you make to your models (adding a field, deleting a model, etc.) into your database schema. They're designed to be mostly automatic, but you'll need to know when to make migrations, when to run them, and the common problems you might run into.

f) Templates: Being a web framework, Django needs a convenient way to generate HTML dynamically. The most common approach relies on templates. A template contains the static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

g) Settings: It is a dotted Python path to a template engine class implementing Django's template backend API. Since most engines load templates from files, the top-level configuration for each engine contains two common settings:

DIRS: defines a list of directories where the engine should look for template source files, in search order.

APP DIRS: tells whether the engine should look for templates inside installed applications. Each backend defines a conventional name for the subdirectory inside applications where its templates should be stored.

h) Static: Django looks for all static files in your apps and collects them wherever you told it to, i.e. the STATIC_ROOT in settings.py gather all static files into a folder called staticfiles in our project root directory.

Implementation

a) Install Django

Django can be installed easily using pip within your virtual environment.

```
pip install django
```

b)Creating a project

```
python django-admin startproject project
```

```
project/
```

```
manage.py
```

```
project/
```

```
__init__.py
```

```
settings.py
```

```
urls.py
```

```
wsgi.py
```

c) The development serve

```
python manage.py runserver
```

d) Creating an app

```
python manage.py startapp app
```

```
app/
```

```
__init__.py
```

admin.py

apps.py

migrations/

__init__.py

models.py

tests.py

views.py

e) Write your first view

app/views.py

```
from django.shortcuts import render
def viewname(request):
    return render(request,'Web.html')
```

f)Write your first URL

app/urls.py

```
from django.conf.urls import url
from . import views
urlpatterns = [
    url(r'$', views.viewname, name='name'),
]
```

The next step is to point the root URLconf at the app.urls module.

project/urls.py

```
from django.urls import include, path
urlpatterns = [
path('app/', include('app.urls')),]
```

g) Creating models

app/models.py

```
from django.db import models
class ModelName(models.Model):
   attributeName = models.CharField(max_length=200)
```

h) Activating models

project/settings.py

```
INSTALLED_APPS = ['app.apps.AppConfig',]
```

i) Create database

```
python manage.py migrate
python manage.py makemigrations polls
```

j) Passing arguments to webpage

```
return render(request, 'Web.html', {'argumentName': argument})
```

To use the data in html

```
{{ argumentName }}
```

k)Building a form in Django

app/forms.py

```
from django import forms
class FormName(forms.ModelForm):
    attributeName = forms.CharField(label='label', max_length=100)
```

app/views.py

```
from .forms import FormName
def get_name(request):
    if request.method == 'POST':
        form = NameForm(request.POST)
        if form.is_valid():
            #Autamatically called when form is supported
```

To save the entries into database

```
form.save()
```

To render the form

```
<form action="/your-name/" method="post">
    {% csrf_token %}
```

```
        {{ form }}
        <input type="submit" value="Submit">
    </form>
```

l) Managing static files (e.g. images, JavaScript, CSS)

Configuring static files

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [
   os.path.join(BASE_DIR, "static"),
   '/var/www/static/',
]
```

To render static files in webpage

```
{% load static %}
<img src="{% static "app/example.jpg" %}" alt="My image">
```

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 INTRODUCTION

The requirements specification is a technical specification of requirements for the software products. It is the first step in the requirements analysis process it lists the requirements of a particular software system including functional, performance and security requirements. The requirements also provide usage scenarios from a user, an operational and an administrative perspective. The purpose of software requirements specification is to provide a detailed overview of the software project, its parameters and goals. This describes the project target audience and its user interface, hardware and software requirements. It defines how the client, team and audience see the project and its functionality.

3.1.1 PURPOSE OF DOCUMENT

This software requirement specification describes all the requirements elicited for "CBIT FORUMS" and is intended to be used by the members examining the project and implementing and verifying the application. Unless otherwise noted all requirements are of high priority and are committed.

3.2 USERS AND THEIR CHARACTERISTICS

An institution can prosper only if the defects can be discovered and solved. The defects can only be discovered if a proper feedback can be taken from the students.Cbit Forums is useful for students to report their issues as well as the management to be able to extract the required information from the students to improve the college.

3.3 SOFTWARE AND HARDWARE REQUIREMENTS

Criterion	Description
OS version	Microsoft® Windows® 7/8/10 (32-bit or 64-bit)
RAM	3 GB RAM minimum, 8 GB RAM recommended
Python	Version 3.4
Packages	django, nltk, scipy, numpy, profanity, gensim, re, matplotlib, wordcloud

4. IMPLEMENTATION

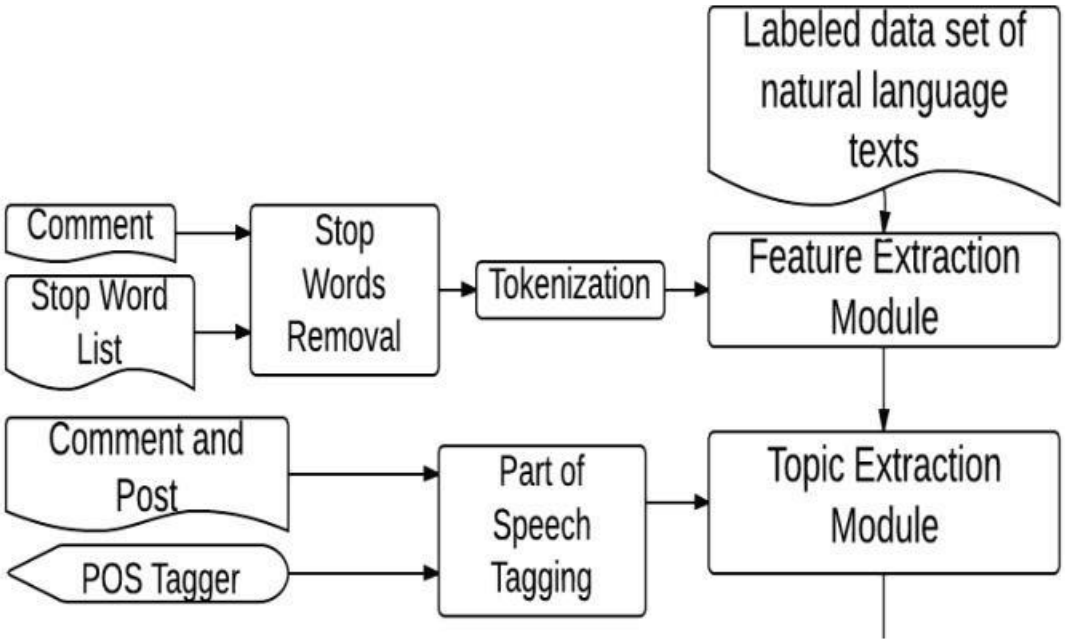
4.1.1 WORKING STAGES OF OUR PLATFORM

Our CBIT DISCUSSION FORUM doesnt work like other discussion forums which are available online.There are three different stages in which it operates:

- Forum
- Discussion display
- Statistics

We will look into each section on how it works and what are its features:

Implementation process:



4.1 figure depicting working stages in project

4.1.2 FORUM

Here in our forum area users can post their issues in any format as they wish ,for example as a suggestion problem, complaint or any other format. Along with the comment they are striving to discuss they even need to mention about the topic.

ALGORITHM INVOLVED ONCLICK SUBMIT:

As users click the submit button an entire background nlp(natural language process takes place we will look into it by stages:
Let us look into it by algorithm we implemented:

a)Whatever the comment is being submitted is being preprocessed as we are using the nltk in python.

b)First step is to check the profanity in the sentence ,as we already came to know about the profanity module in nltk,we need to provide a word list in the a document format and import it so that when use the profanity module on the sentences it will check from that wordlist.
If any of the word is found bad then the further preprocessing is stopped exiting the process as bad words detected.

c)If any type of profanity is not found then it will continue to remove the stop words as they do not contribute to the topic detection and similarity.

d) Then we tokenize the sentences using the word tokenizer and the sentence tokenizer.
e)We need to find the parts of speech of the words using pos_tag as to arrive at the topic detection we need nouns+nouns else nouns+adjectives.Mostly the topic can be detected if we preprocess further on these words.

f)Then we arrive at a stage to find out the ngrams :

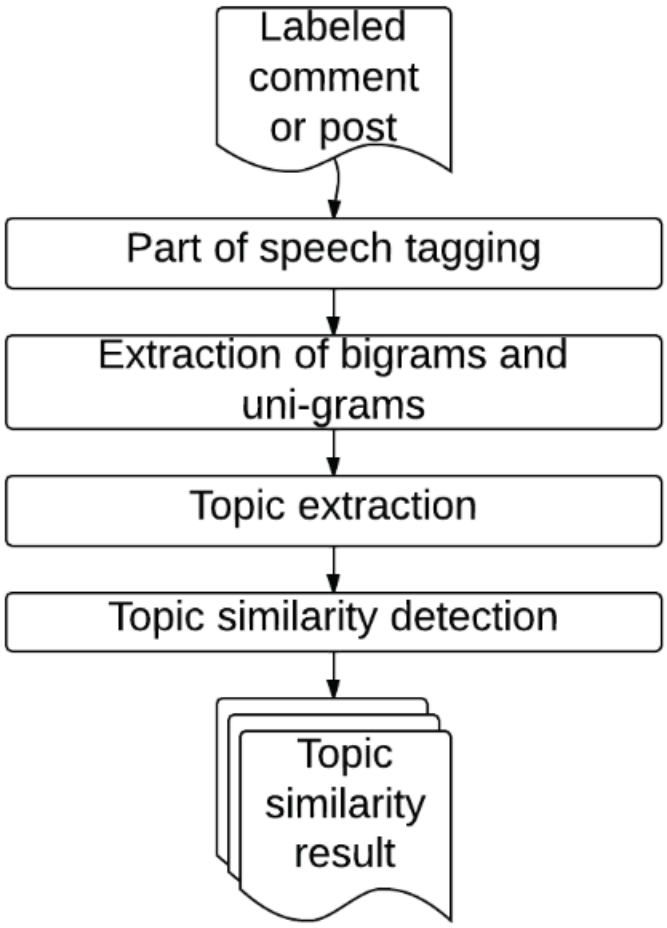
Ngrams are nothing but the dividing the sentences to two or more word together ,they are called bigrams ,trigrams respectively. Grouping word with their adjacent word gives result with more accuracy.

g)LDA(Latent Dirichlet Allocation) is applied in this part of preprocessing:

Once you provide the algorithm with the number of topics, all it does it to rearrange the topics distribution within the documents and keywords distribution within the topics to obtain a good composition of topic-keywords distribution. A topic is nothing but a collection of dominant keywords that are typical representatives. Just by looking at the keywords, you can identify what the topic is all about.
The following are key factors to obtaining good segregation topics:

i. The quality of text processing.
ii. The variety of topics the text talks about.
iii. The choice of topic modeling algorithm.
iv. The number of topics fed to the algorithm.
v. The algorithms tuning parameters.

Whatever the output comes after this is displayed in the discussion section'



4.2 figure depicting process of LDA

4.1.3 DISCUSSION :

All the discussion which are displayed in this section will be taken to the next part of the natural language processing.

Here in this section sentiment analysis is done where we try to find out the positivity and negativity in the comments users generally post so we can come to conclusions on what they are mostly talking.

Classification is done using several steps: training and prediction.

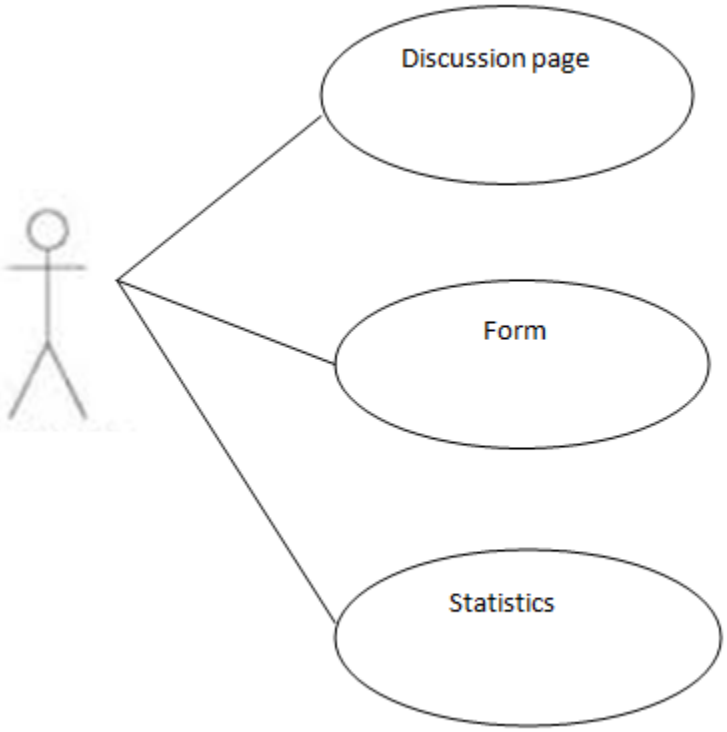
The training phase needs to have training data, this is example data in which we define examples. The classifier will use the training data to make predictions. We start by defining 3 classes: positive, negative and neutral.

4.1.4 STATISTICS

Next step involves where we create a word cloud and a bar graph which are displayed in our statistics section

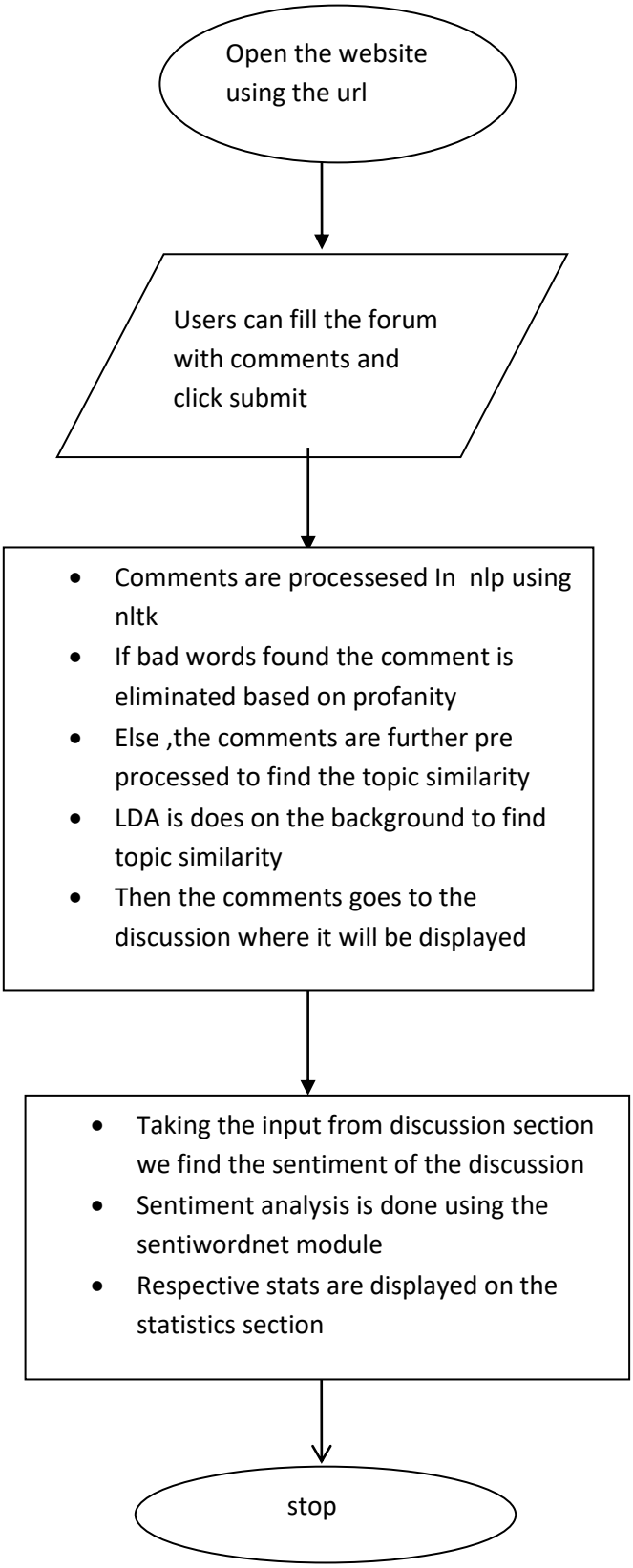
Showing and saving the statistics every time we update the discussion is an important part, accordingly we need make the django and web page connections correctly as we explained in the technologies part of the work we need to done to do all the render() and the other views and models.

4.2 USE CASE DIAGRAM



4.3 figure depicting use case diagram

4.3 PROJECT FLOW



5.RESULTS

5.1 INTRODUCTION

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

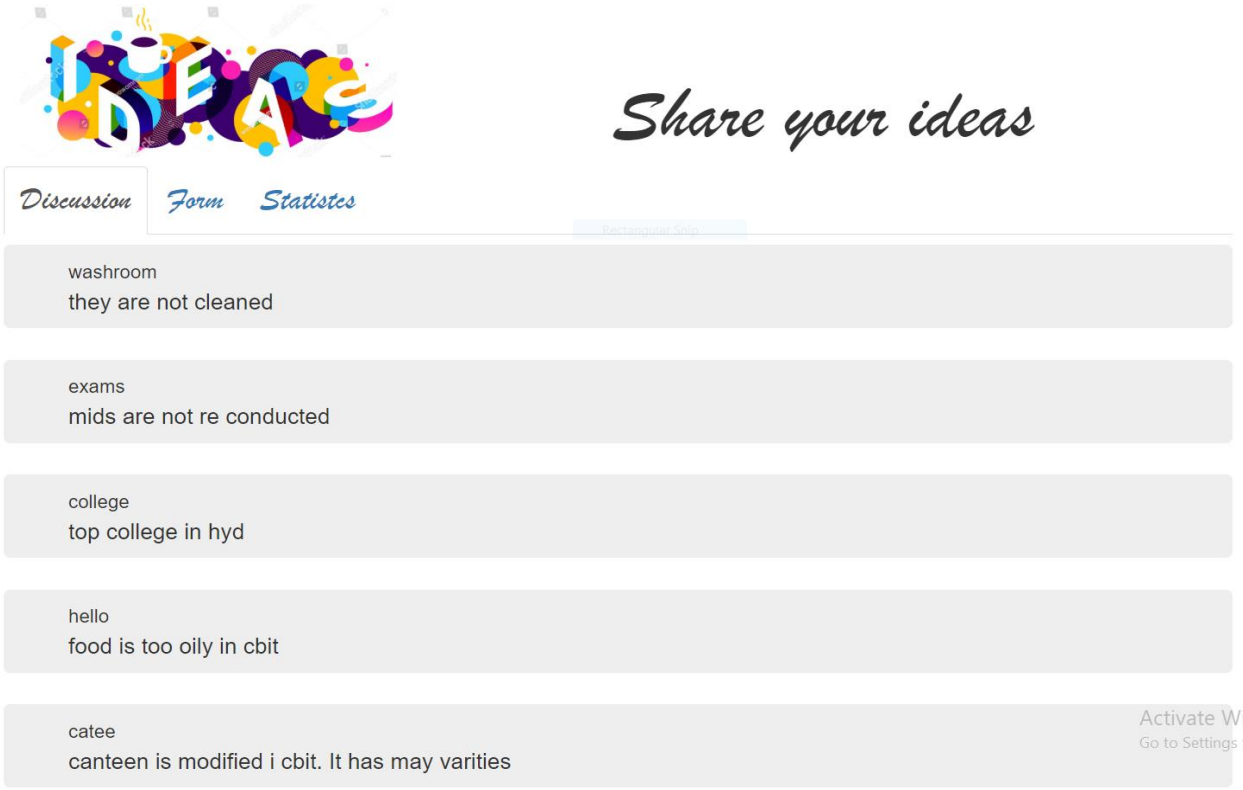
A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

5.2 TESTING OBJECTIVES

The main objective of performance testing is designed to test whether display is as expected and whether the source code is functioning properly or not. As the test results are gathered and evaluated they begin to give a qualitative indication of the reliability of the code. If proper output is not obtained, the overall quality of the code is questioned. If, on the other hand, all the results which are not successful, are encountered, and are easily modifiable, then the following conclusion can be made: The tests are inadequate as the requirements mentioned are not compatible. The testing includes:

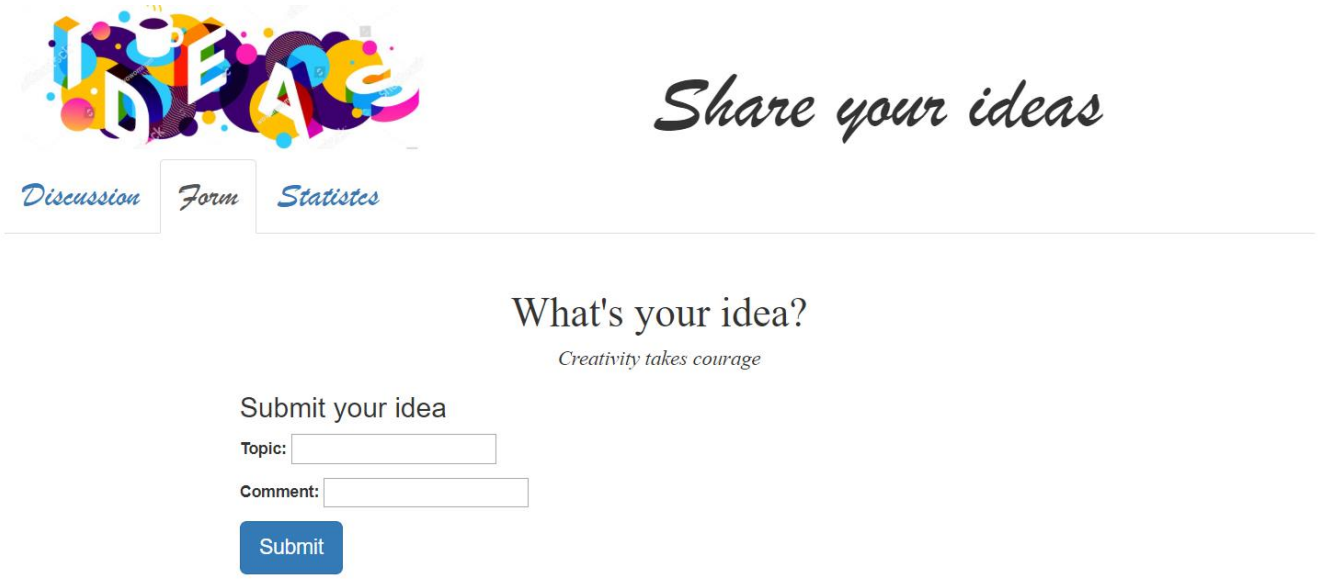
- Checking whether the information is displayed or not.
- Checking whether files are correctly linked and opened.
- Checking whether the required data is displayed correctly or not by checking through android emulator or by mobile phones.

5.3. OUTPUT SCREENS



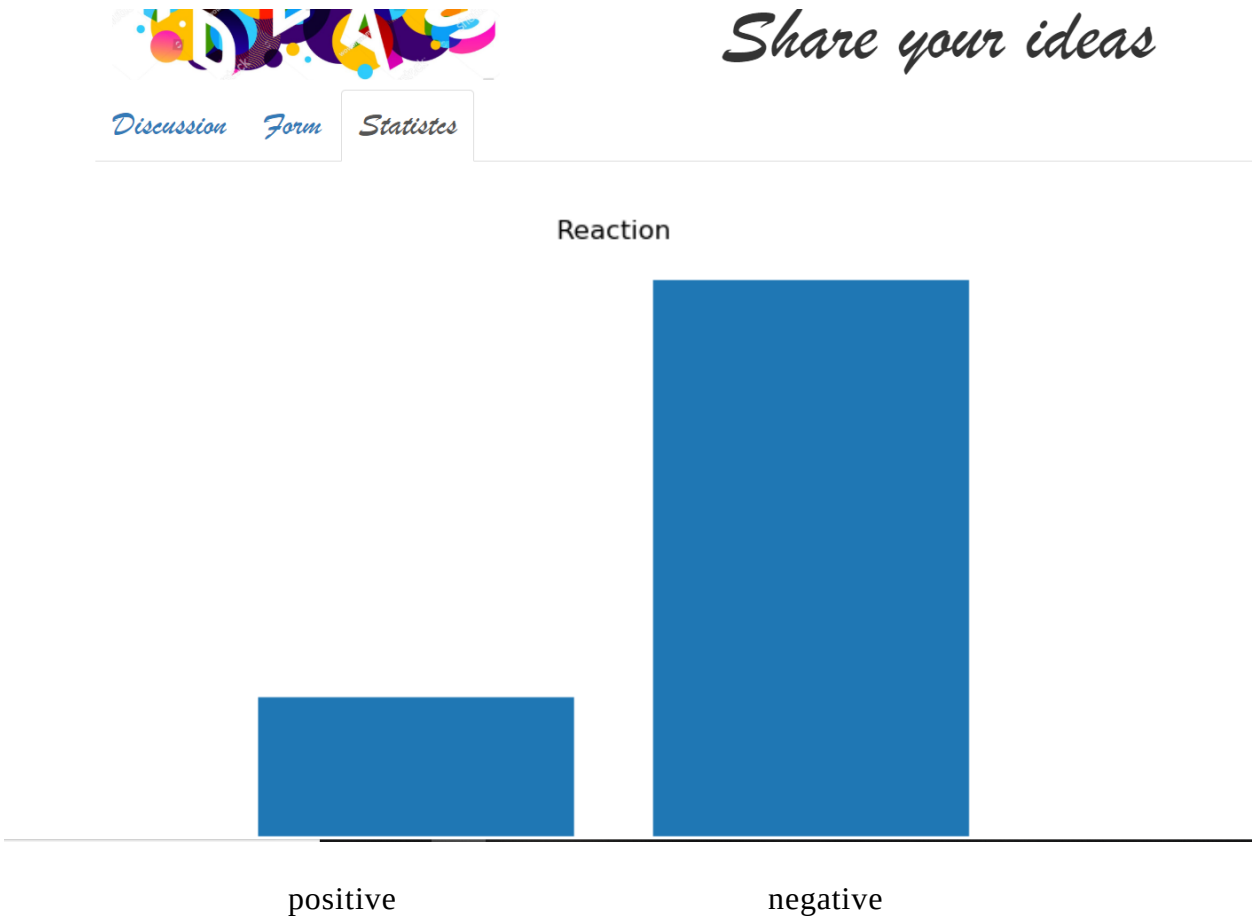
5.1 Discussion page in our website

This is our main discussion page where it is the first visible page to the users who access the website. Comments which are submitted by different users after entire pre processing are displayed in this page, Discussions which are displayed after pre processing are then taken to perform sentiment analysis.



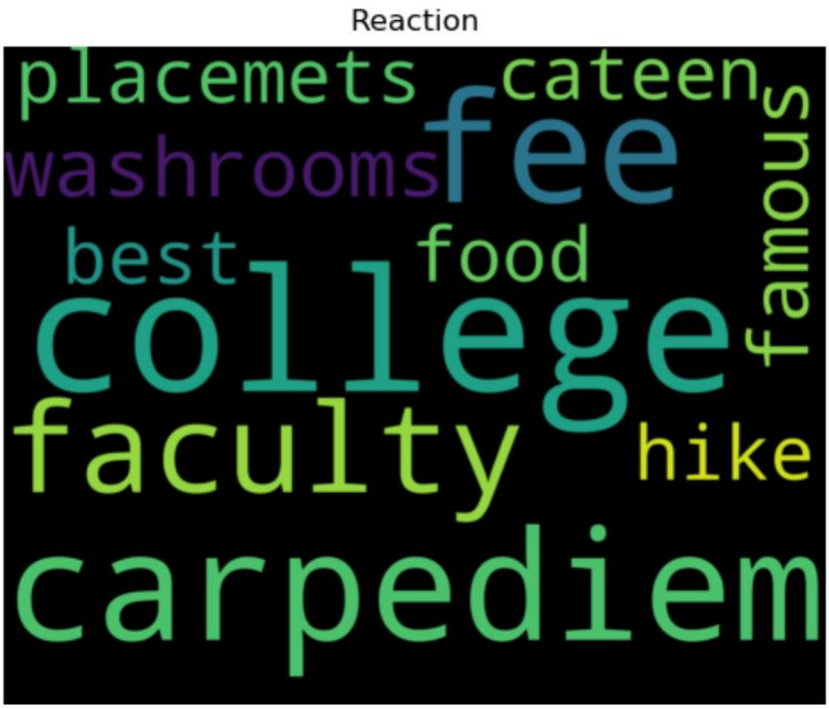
5.2 Forum page in our website

Forum page is the second section in the website where users can speak about any topic and give their suggestions ,comments, complains etc that to anonymously after submitting the comments they will be pre processed and LDA algorithm is performed an we obtain topics mostly talked and an basic overview is obtained.



5.3 figure depicting barplot of the sentiment analysis output

This barplot is displayed in the statistics section where the comments after pre processing are sent to sentiment analysis where we get the positivity and negativity of comments and then we show it pictorially in the following barplot



5.4 word cloud of sentiment analysis output

This pictorial representation of the sentiment analysis in form of word cloud gives the majorly discussed topics and taking a look on this gives the entire view of the discussion going on currently

6. CONCLUSION AND FUTURE WORK

Our project "CBIT Forums" has been developed completely in such a way that the application cannot be used for any bad practices. The students can write comments and submit them ananymously without the fear of being targetted. But the comment is only recorded if it is appropriate and does not contain any fowl language. In this way only the required information can be categorised. The weekly basis major problem is displayed so that the management can take required action and keep the institute growing.

In the future time we intend to make this application an official platform to discuss the issues. We would like to add the email notification functionality which is mailing the appropriate personnel about the problem that is trending and needs to be solved as early as possible. We also want to block the unwanted users who repeatedly try to comment inappropriately. This makes sure that the platform is free from any innapropriate content and is used only in the positive manner

7. BIBLIOGRAPHY

Research paper:

https://ieeexplore.ieee.org/document/6936976/

Websites:

https://docs.djangoproject.com/en/2.1/

https://www.analyticsvidhya.com/blog/2017/01/ultimate-guide-to-understand-implement-natural-language-processing-codes-in-python/

https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24

https://matplotlib.org/tutorials/introductory/pyplot.html

https://www.geeksforgeeks.org/generating-word-cloud-python/

http://nlp.stanford.edu/software/corenlp.shtml.

Books:

NLP AT WORK by sue knight