

A Project Plan  
On  
**Automatic Highlighter of Lengthy Legal Documents**  
Submitted to  
Galgotias University



In partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Anish

Enrollment no.=1613101135

Admission no.=16SCSE101250

Under the guidance of

**Dr. Sasikumar.P**

**Declaration**

I, Anish (Enrol no. 1613101135,admsn no. 16SCSE101250 student of B.Tech (CSE) hereby declare that the project titled “**Automatic highlighter of lengthy legal documents** ” which is submitted by us to Department of Computer Science and Engineering, in partial fulfillment of requirement for the award of the degree of Bachelor of Technology in CSE has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition.

**Date:** 04/05/2020

**Anish**

**1613101135**

**16SCSE101250**

## **Certificate**

On the basis of declaration submitted by **Anish (1613101135)**, students of B. Tech Computer Science and Engineering, I hereby certify that the project titled " Automatic highlighter of lengthy legal documents " which is submitted to Department of Computer Science and Engineering, in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in 2020, is an original contribution with existing knowledge and faithful record of work carried out by him/them under my guidance and supervision.

To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

## **Acknowledgement**

We would like to acknowledge the efforts of all the individuals whose kind support and guidance made this project possible .I would like to extend our sincere thanks to all of them.

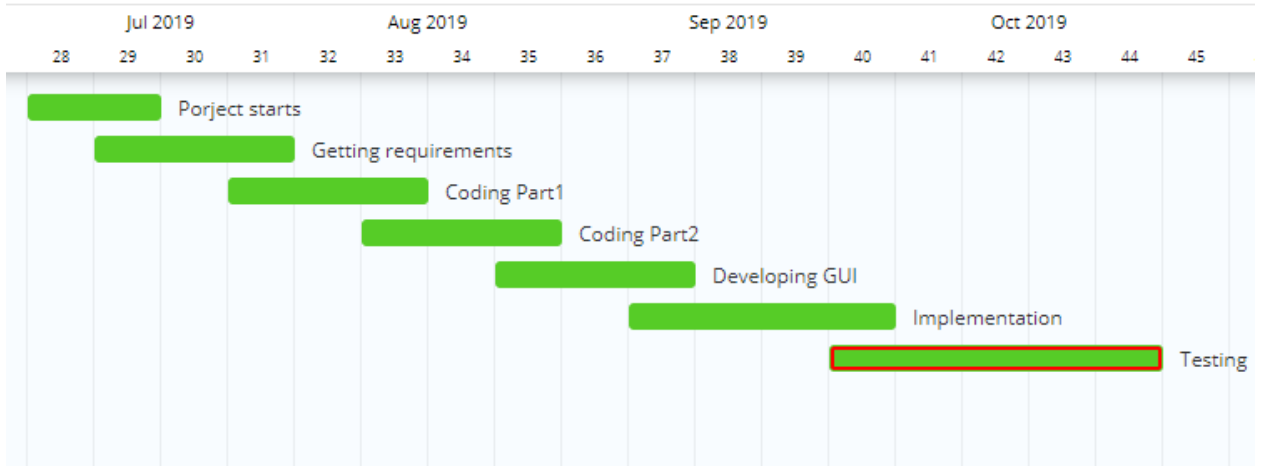
I am highly indebted to **Dr.Sasikumar.P** for taking us under his guidance and supervision. We are grateful that we shared his vast ocean of knowledge and experience of many years with us .

I would also like to extend my gratitude towards my family and friends who supported me throughout the project.

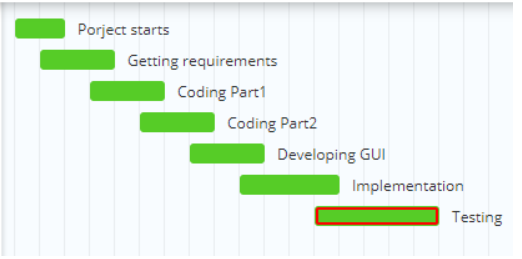
**ABSTRACT**

This project addresses the problem of readers to read a entire lengthy legal document completely which consumes a lot of time and effort. This project highlights the important aspects of a document to provide the readers at their ease to read only the important points of the document. This helps a reader to understand a document properly with ease of effort and consumes only a little time. A reader who doesn't even have a technical knowledge of the document may also get an overview of the project.

## **Gantt chart**



0	✔	Project starts	Rohit Rathore	-	11/Jul	5d	15/Jul	100%	
1	✔	Getting requirements	Nishita Singh	-	16/Jul	16d	31/Jul	100%	
2	✔	Coding Part1	Nishita Singh	-	01/Aug	15d	15/Aug	100%	
3	✔	Coding Part2	Rohit Rathore	-	15/Aug	17d	31/Aug	100%	
4	✔	Developing GUI	Nishita Singh	-	01/Sep	12d	12/Sep	100%	
5	✔	Implementation	Rohit Rathore	-	13/Sep	18d	30/Sep	100%	
6	✔	Testing	Nishita Singh	-	01/Oct	31d	31/Oct	100%	



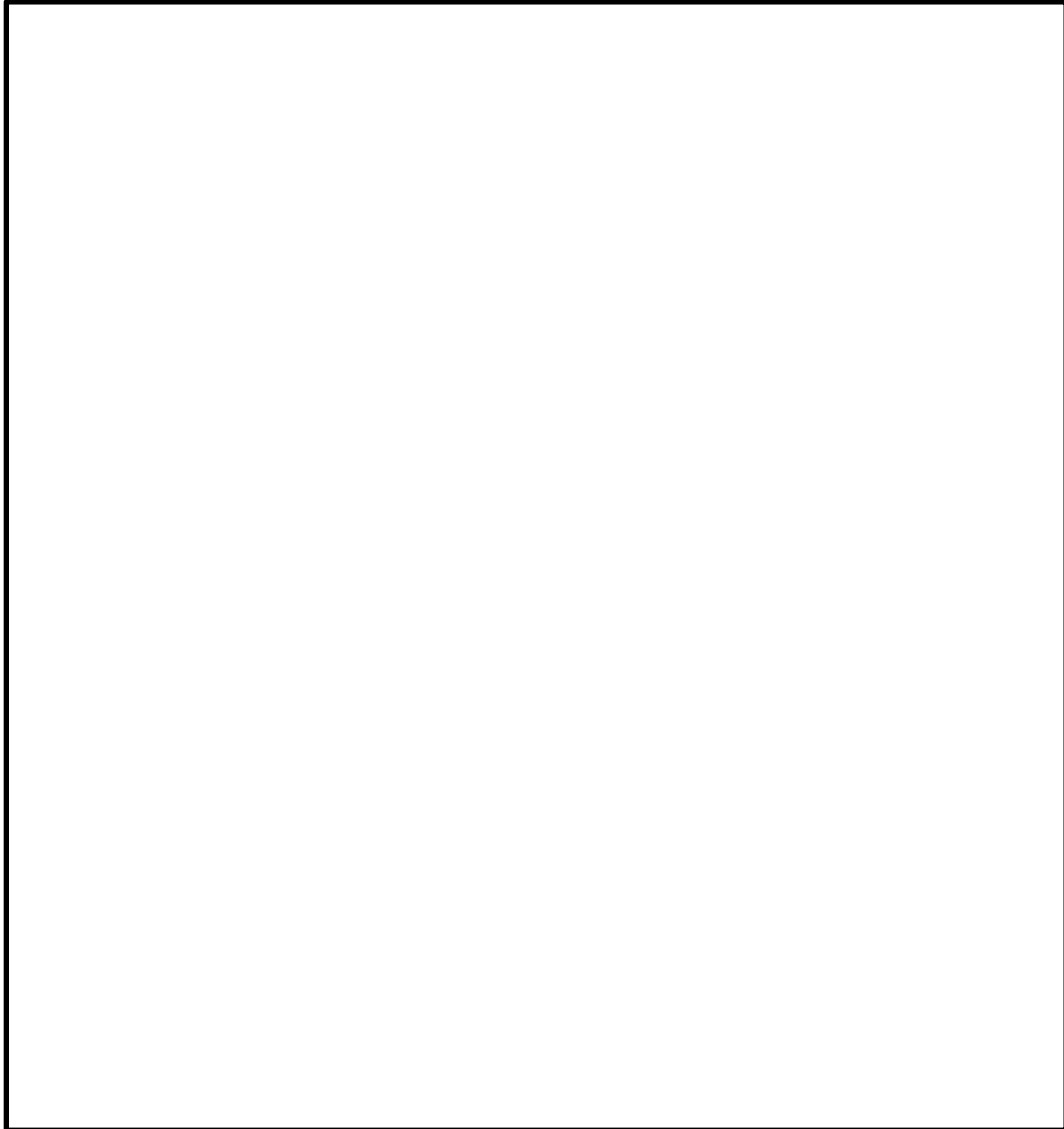
## **TABLE OF CONTENT**

<b><u>TOPICS</u></b>	<b><u>PAGE</u></b>
<b>1.Introduction</b>	<b>9-17</b>
<b>1.1 Text Summarization</b>	
<b>1.2 LDA</b>	
<b>1.3 Word embeddings (Word2Vec)</b>	
<b>2. Objective</b>	<b>18</b>
<b>3. Literature Review</b>	
<b>3.1 Technique Used</b>	<b>18-19</b>
<b>3.2 Algorithm Used</b>	<b>19-21</b>
<b>4. Phases of Project</b>	<b>21-27</b>
<b>5. Training Output, Test &amp; Result</b>	<b>28-40</b>
<b>6. Conclusion</b>	<b>41</b>
<b>7. Future Work</b>	<b>42</b>
<b>8. References</b>	<b>43</b>

## LIST OF FIGURES

S. No.	Table Details	Page No.
1.2	Illustration of LDA input/output workflow	13
4.1(i)	Making Corpus of All the Data Sets	22
4.1(ii)	Making the LDA Model	23
4.2(i)	Training Word2Vec Model	24
4.3(i)	Clustering the Test Document	25
4.4(i)	Anomaly Detection	27
4.5(i)	DOCX file making	28
4.5(ii)	Converting to PDF	29
5.1(i)	Result of LDA trained model	30
5.2(i)	Vector representation of word 'apple'	31
5.2(ii)	Vector representation of word 'crime'	32
5.2(iii)	Vector representation of word 'agreement'	33
5.2(iv)	Vector representation of word 'license'	34
5.2(v)	Vector representation of word 'liability'	35
5.2(vi)	Vector representation of word 'law'	36
5.2(vii)	Vector representation of word 'customer'	37
5.2(viii)	Vector representation of word 'software'	38
5.3(i)	Test File One with No Irrelevant Sentence	39
5.3(ii)	Result Corresponding to above Test File	40
5.3(iii)	Test File Two with Irrelevant Sentences, Highlighted One	41

**COMMENT BY EXTERNAL EXAMINER**



**Name of the External Examiner :**

**Signature:**



# CHAPTER 1

## INTRODUCTION

### A. Background

Authoritative reports are known for being extensive. As far as anyone is concerned, a few classes of authoritative reports contain copied data that don't require our consideration.

Notwithstanding, physically separating non-copy data from reports requires extensive measure of exertion. In this manner, one want to utilize AI calculations to get super sentences for us. In this paper, the proposed number of calculations that channels out duplicate information and returns useful statistics to the client. One can put together a scholar which can stamp superb pieces of an authoritative document for manual research.

The analyzing manner carries degrees. At the critical degree, one pick some authoritative evaluations that incorporate normal examples, for example programming client understandings, to border a mastering base for the mentor. At that point run LDA model on these reviews. The LDA version will return us with a whole lot of basic subjects over the records base. At the following stage, take every other little bit of authoritative record due to the fact the take a look at. First expel ordinary challenge words from the test archive to assemble contrasts among sentences, then use Word2Vec to change over sentences into vectors. In the wake of making the element region, run Agglomerative Clustering and Local Outlier Factor (LOF) calculations on the detail vectors to distinguish unique sentences in the given document.

At last, use PCA and t-SNE to count on our very last results.

## **B.**

### **1.1 Text Summarization**

Content outline is the way toward refining the most significant data from a source (or sources) to deliver a condensed form for a specific client (or clients) and errand (or undertakings).

People are commonly very great at this undertaking as we have the ability to comprehend the significance of a book record and concentrate notable highlights to condense the reports utilizing our very own words. In any case, programmed strategies for content rundown are vital in this day and age where there is an excess of information and absence of labor just as time to translate the information. There are numerous reasons why Automatic Text Summarization is helpful:

1. Rundowns diminish understanding time.
2. When looking into archives, Synopses make the choice manner less difficult.
3. Programmed outline improves the feasible of ordering.
4. Programmed outline calculations are much less one-sided than human summarizers.
5. Customized outlines are valuable being noted noting frameworks as they give customized information.
6. Utilizing programmed or self-loader rundown frameworks empowers commercial enterprise theoretical executives to expand the quantity of content material files they can system.

Synopsis frameworks often have extra evidence they could use so as to show the most significant subjects of file(s). For example, while outlining net journals, there are talks or comments coming after the blog access which can be terrific wellsprings of records to determine out which parts of the weblog are simple and captivating.

In logical paper synopsis, most of the statistics, for instance, referred to papers and collecting facts which can be applied to understand significant sentences inside the first paper.

A repetitive subject in NLP is to see enormous corpus of writings through themes extraction. Regardless of whether you investigate clients' online audits, items' portrayals, or content entered in search bars, understanding key themes will consistently prove to be useful.

## 1.2 LDA

LDA (*short for Latent Dirichlet Allocation*) is an unaided AI model that accepts reports as information and discovers subjects as yield. The model likewise says in what rate each record discusses every subject.

A subject is spoken to as a weighted rundown of words. A case of a point is demonstrated as follows:

blossom \* 0,2 | rose \* 0,15 | plant \* 0,09 |...

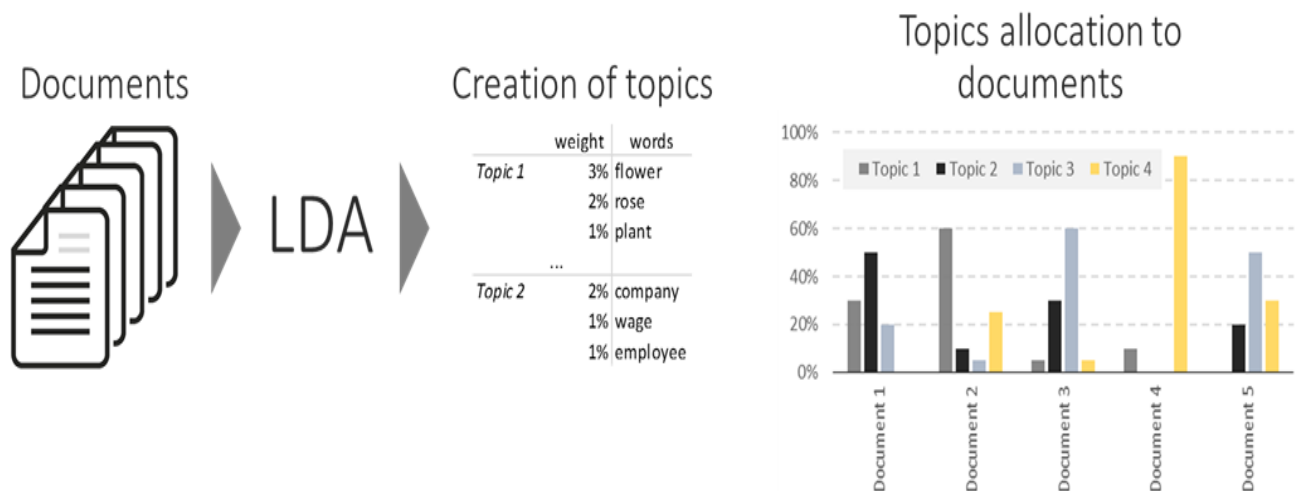


Fig 1.2(i) Illustration of LDA input/output workflow

There are 3 fundamental parameters of the model:

1. the quantity of themes
2. the quantity of words per theme
3. the quantity of themes per record

As a general rule, the last two parameters are not actually planned like this in the calculation, yet I like to adhere to these disentangled renditions which are more obvious.

### IMPLEMENTATION: -

To execute the LDA in Python, I utilize the bundle gensim.  
The parameters demonstrated beforehand are:

1. the quantity of themes is equivalent to num\_topics
2. the [distribution of the] number of words per theme is dealt with by estimated time of arrival
3. the [distribution of the] number of themes per report is taken care of by alpha

Fundamental points of interest of LDA:

1. It's quick -Utilize the %time direction in Jupyter to confirm it. The model is typically quick to run. Obviously, it relies upon your information. A few components can hinder the model:

2. Long archives -Huge number of archives

Huge jargon size (particularly in the event that you use n-grams with an enormous n)

3. It's natural – Displaying subjects as weighted arrangements of words is a straightforward estimation yet an exceptionally natural methodology in the event that you have to decipher it. No installing nor shrouded measurements, just sacks of words with loads.

4. It can anticipate subjects for new concealed reports -When the model has run, it is prepared to apportion subjects to any report. Obviously, if your preparation dataset is in English and you need to foresee the themes of a Chinese report it won't work. In any case, if the new records have a similar structure and ought to have pretty much similar points, it will work.

## Main disadvantages of LDA

1. Heaps of calibrating -On the off chance that LDA is quick to run, it will give you some issue to get great outcomes with it. That is the reason knowing ahead of time how to adjust it will truly support you.

2. It needs human elucidation - Subjects are found by a machine. A human need to name them so as to show the outcomes to non-specialists individuals.

3.You can't impact subjects - Realizing that a portion of your records talk about a theme you know, and not discovering it in the points found by LDA will disappoint. What's more, there's no real way to state to the model that a few words ought to have a place together. You need to sit and trust that the LDA will give you what you need.

LDA stays one of my preferred models for subject's extraction, and I have utilized it numerous activities. Be that as it may, it requires some training to ace it. That is the reason I caused this article with the goal that you to can hop over the obstruction to section of utilizing LDA and use it easily.

## 1.3 Word embeddings (Word2Vec)

### 1.3.1 Text as numbers:

AI fashions take vectors (types of numbers) as information. When operating with content material, the primary issue we have to do concoct a way to trade over strings to numbers (or to "vectorize" the content material) before maintaining it to the version. In this section, one takes a gander three of the procedures for doing the same.

### 1.3.2 One-hot encodings

As a primary notion, we may also "one-hot" encode every phrase in our jargon. Consider the sentence "The pussycat sat at the tangle". The jargon (or incredible words) on this sentence is (tom cat, tangle, on, sat, the). To talk to each phrase, one can make a 0 vector with length equivalent to the jargon, at that point vicinity a one inside the document that relates to the word.

To make a vector that consists of the encoding of the sentence, one may want to then link the only-warm vectors for each word.

Key factor: This method is wastage. A one-hot encoded vector is meager (which means that, most indicates are zero). Envision we've 10,000 phrases inside the jargon. To one-hot encode every phrase, one would make a vector where ninety nine.99% of the additives are 0.

### 1.3.3 Encode each word with an exceptional number

The other approach one can also strive is to encode every word using an brilliant wide variety. Proceeding with model above, we should allot 1 to "feline", 2 to "tangle, and so on. We ought to then encode the sentence "The feline sat on the tangle" like thick vector [5, 1, 4, 3, 5, 2]. This approach is powerful. Rather than a scanty vector, we presently have a thick one (wherein all additives are complete).

Some of the drawbacks to this system :

The whole number encoding is discretionary (it doesn't catch any connection between words).

A range encoding may be attempting for a model to decipher. A direct classifier, as an example, learns a solitary load for each detail. Since there is no connection between the likeness of any phrases and the similitude of their encodings, this detail weight blend isn't always significant.

### 1.3.4 Word embeddings

Word embeddings give us a way to use a green, dense illustration wherein similar phrases have a comparable encoding. Importantly, we do no longer need to specify this encoding with the aid of hand. An embedding is a dense vector of floating factor values (the length of the vector is a parameter you specify). Instead of specifying the values for the embedding manually, they may be trainable parameters (weights learned by means of the model for the duration of training, in the equal way a version learns weights for a dense layer). It is common to peer word embeddings which are 8-dimensional (for small datasets), as much as 1024-dimensions while operating with large datasets. A better dimensional embedding can seize great-grained relationships among phrases, however takes more statistics to analyse.

### 1.3.5 Make a straightforward model

One will utilize the Keras Sequential API to signify our model. For this example, it's far a "Persistent sack of phrases" style model.

1. Next the Embedding layer takes the complete wide variety encoded jargon and looks into the installing vector for every phrase-file. These vectors are observed out because the model trains. The vectors upload a measurement to the yield exhibit. The subsequent measurements are: (bunch, succession, implanting).

2. Next, a GlobalAveragePooling1D layer restores a set-period yield vector for each version by averaging over the association measurement. This permits the model to deal with contribution of variable length, in the least tough way that is to be had.

3. This constant-period yield vector is channelled via a completely related (Dense) layer with sixteen concealed gadgets.

4. The remaining layer is thickly associated with a solitary yield hub. Utilizing the sigmoid enactment paintings, this worth is a buoy someplace in the range of zero and 1, speak me to a likelihood (or truth degree) that the survey is sure.



### 1.3.6 Visualize the embeddings

To imagine our embeddings, we are able to switch them to the installing projector.

Open the Embedding Projector (this may likewise run in a nearby Tensor Board example). Snap on "Burden facts". Transfer the two statistics we made above: `vecs.Tsv` and `meta.Tsv`. The embeddings you have organized will currently be proven. You can test for words to locate their nearest buddies. For instance, have a pass at scanning for "lovely". You may also see pals like "splendid".

## **CHAPTER 2**

### **OBJECTIVE**

To propose a lot of calculations that channels out copy data and returns helpful data to the client. We can prepare a student that can stamp exceptional pieces of an authoritative record for manual investigation. We need to build your own highlights package in Python using a simple approach. In order to save the readers time and effort this project will help the users to automatically highlight important parts of the document for a quick go through. This project will also generate different color highlights and classify data into the level of importance. More important sentences will be highlighted as red and less as blue and those which are irrelevant will be left into black.

## CHAPTER 3

### LITERATURE REVIEW

#### 3.1 Technique Used

Two specific arrangements of highlight extraction and determination procedures.

To begin with, use LDA to build everyday subject matter phrases over the preparation set of information. The element extraction interior LDA is completed via accepting all facts as a solitary corpus and in a while by way of making use of the variational deduction approach mentioned in the first paper.

Subsequent to walking LDA on each one of the reports, one get a rundown of topic phrases. A point is a rundown of words, for example trademark, administrations, might also, use, utility, knowledge, content. It is visible that those problem words do not add to the "electricity" of a sentence.

Along these traces, the factor are expeled phrases from the take a look at file.

Second, for the check document, Word2Vec is used to make an interpretation of a word into a vector. Word2Vec is a two-layer neural system. It accepts content corpus as records and its yield is lots of highlight vectors for words in that corpus. For every phrase, we create one hundred highlights. These phrase vectors think about numeric obligations on words.

For instance, the vector of "king" included by using the vector of "ladies" would restore the vector of "queen". In the wake of going for walks Word2Vec, we building up a mapping from phrases to vectors. It is predicted that the thing vector of a sentence is the overall of highlight vectors of the vast wide variety of words on this sentence. At that point run bunching calculations at the vector portrayals of the great variety of sentences.

## 3.2 Algorithms Used

EM calculation is utilised to end up acquainted with the shrouded parameters of the LDA version. At that factor utilize Agglomerative Clustering and LOF to grow to be familiar with the dispersions of high-quality sentences within the archive.

### 1) EM for estimating LDA parameters

The LDA model uses a word  $w$  as an essential unit. A file is a grouping of  $N$  words signified with the aid of  $w = (w_1, \dots, w_N)$ . A corpus contains  $M$  documents is signified by  $D = w_1, \dots, w_M$ .

### 2) Agglomerative Clustering

For any other test report, we to begin with get the detail vector of every sentence. We at that factor make use of Agglomerative Clustering to element the report. We initially decide that we want to have  $N$  organizations. We at that factor placed each sentence into its very personal bunch, and iteratively blend the closest groups until simply  $N$  corporations continue to be. We at that factor take a gander at these corporations. On the off risk that there exists a gaggle that incorporates simply one sentence, at that factor we recognize that this sentence is a unique one, as it's far in no way converged into one-of-a-kind organizations.

### 3) Local Outlier Factor

The close by anomaly aspect relies upon on an idea of a neighborhood thickness, in which territory is given with the aid of closest neighbors, whose separation is utilized to appraise the thickness. By looking on the close by thickness of an editorial to the neighborhood densities of its associates, you'll distinguish locales of comparative thickness, and focuses which have a significantly decrease thickness than their friends. These are regarded as exception.

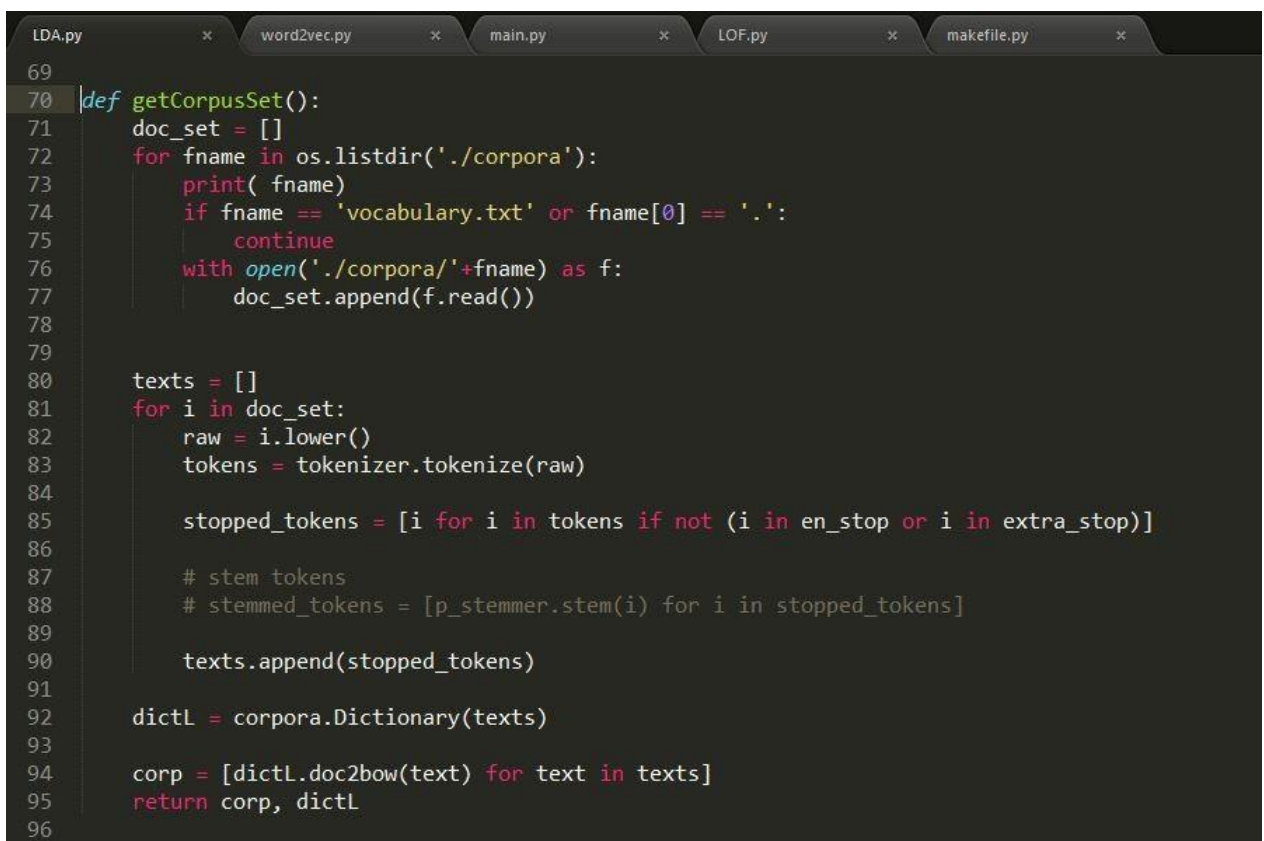
## CHAPTER 4

### PHASES OF PROJECT

#### 4.1 Phase 1-Learning Phase

Legal documents are regarded for being lengthy. To our know-how, some categories of criminal documents include duplicated statistics that do not require our attention. However, manually extracting non-reproduction records from documents calls for widespread amount of attempt. Thus, we need to use machine studying algorithms to choose up unordinary sentences for us.

At the first phase, we choose a few legal documents that contain common styles, eg, Software program user agreements, to shape a expertise base for the instructor. We then run LDA version on those documents. The LDA model will go back us with a set of common subjects throughout the expertise base. At the second section, we take a new piece of felony report because the test sample. We first eliminate common topic phrases from the take a look at document to growth differences between sentences.

A screenshot of a code editor with a dark background and light-colored text. The editor has several tabs at the top: 'LDA.py', 'word2vec.py', 'main.py', 'LOF.py', and 'makefile.py'. The main window displays Python code for a function named 'getCorpusSet()'. The code starts at line 69 and ends at line 96. It defines a list 'doc\_set' and iterates over files in the './corpora' directory. It filters out 'vocabulary.txt' and files starting with a dot. For each file, it reads the content, tokenizes it, removes stop words, and stems the remaining tokens. The processed tokens are added to a 'texts' list. Finally, it creates a 'dictL' dictionary and returns a list of document-bow pairs and the dictionary.

```
69
70 def getCorpusSet():
71     doc_set = []
72     for fname in os.listdir('./corpora'):
73         print( fname)
74         if fname == 'vocabulary.txt' or fname[0] == '.':
75             continue
76         with open('./corpora/'+fname) as f:
77             doc_set.append(f.read())
78
79
80     texts = []
81     for i in doc_set:
82         raw = i.lower()
83         tokens = tokenizer.tokenize(raw)
84
85         stopped_tokens = [i for i in tokens if not (i in en_stop or i in extra_stop)]
86
87         # stem tokens
88         # stemmed_tokens = [p_stemmer.stem(i) for i in stopped_tokens]
89
90         texts.append(stopped_tokens)
91
92     dictL = corpora.Dictionary(texts)
93
94     corp = [dictL.doc2bow(text) for text in texts]
95     return corp, dictL
96
```

Fig. 4.1(i) Making Corpus of All the Data Sets

Then run LDA version on the corpus we fashioned. The LDA model will go back us

with a fixed of not unusual topics throughout the understanding base. At the second one section, we take a new piece of legal record as the check pattern. We first get rid of commonplace subject matter phrases from the test file to boom variations between sentences.

```

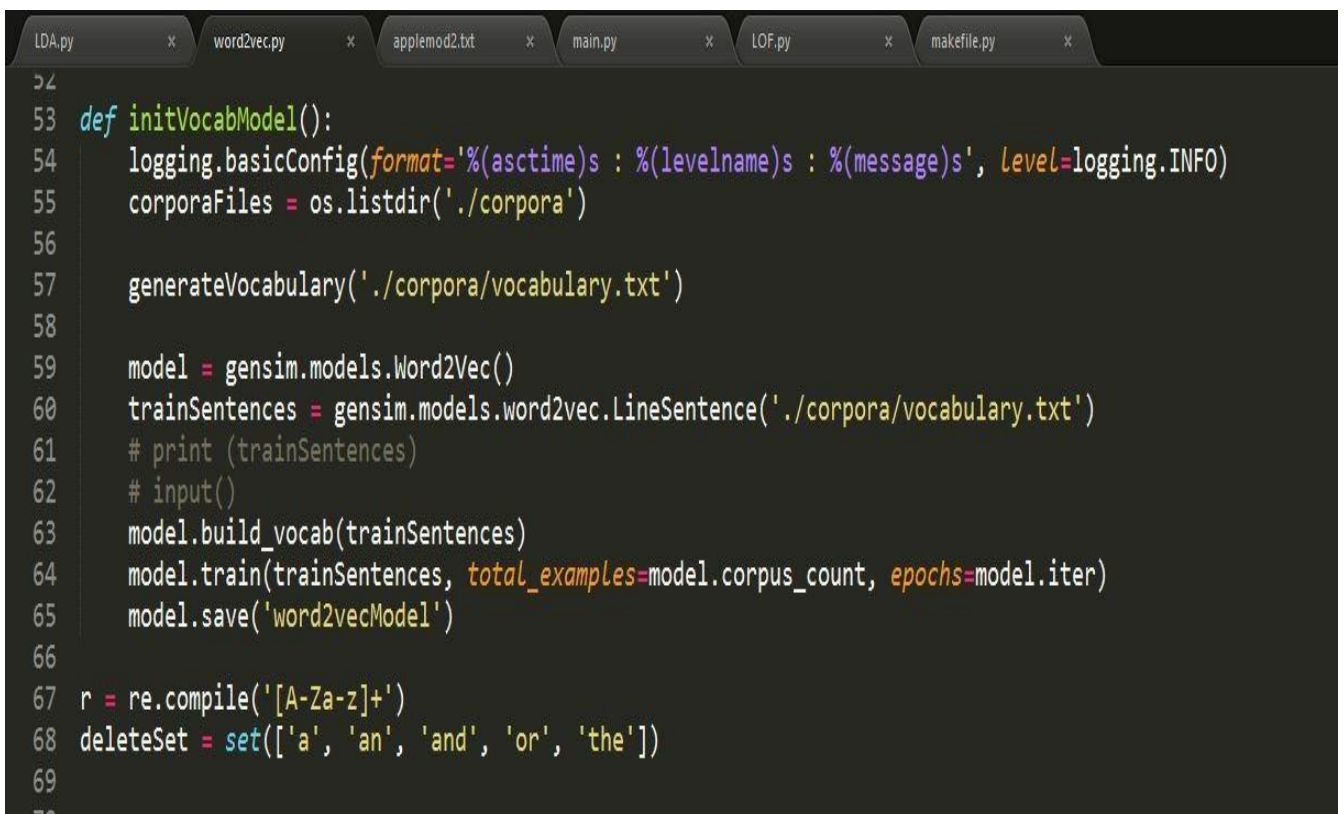
27
28 def findSuitableNumTopics(bowCorpus, trainDict):
29     parameterL = range(3, 20, 1)
30     grid = dict()
31     perWordGrid = dict()
32     trainSize = int(round(len(bowCorpus)*0.8))
33     trainInd = sorted(random.sample(range(len(bowCorpus)), trainSize))
34     testInd = sorted(set(range(len(bowCorpus))) - set(trainInd))
35     trainCorp = [bowCorpus[i] for i in trainInd]
36     testCorp = [bowCorpus[i] for i in testInd]
37
38     numOfWords = sum(cnt for document in testCorp for _, cnt in document)
39     for param in parameterL:
40         grid[param] = list()
41         perWordGrid[param] = 0
42         print( ' >>>>>>>> starting using # topics = ', param)
43         model = gensim.models.LdaModel(corpus=trainCorp, id2word=trainDict, num_topics=param, iterations=10)
44         perplex = model.bound(testCorp)
45         print( 'total perplexity = ', perplex)
46         grid[param].append(perplex)
47
48         perWrdPerplex = np.exp2(-perplex / numOfWords)
49         print( 'per word perplexity = ', perWrdPerplex)
50         grid[param].append(perWrdPerplex)
51         if param == 3:
52             perWrdPerplex += 102
53         elif param == 4:
54             perWrdPerplex += 40
55         perWordGrid[param] = perWrdPerplex
56
57
58     df = pandas.DataFrame(grid)
59     ax = plt.figure(figsize=(7, 4), dpi=300).add_subplot(111)
60     df.iloc[1].transpose().plot(ax=ax, color="#254F09")
61     plt.xlim(parameterL[0], parameterL[-1])
62     plt.ylabel('Perplexity')
63     plt.xlabel('topics')
64     plt.title('')
65     plt.savefig('perplexityResult.png', format='png', bbox_inches='tight', pad_inches=0.1)
66     # plt.show()
67     return grid, perWordGrid
68

```

Fig 4.1(ii) Making the LDA Model

## 4.2 Phase 2-Training the Word2Vec Model

We then use Word2Vec to convert sentences into vectors. Word2Vec translate a phrase into a vector. Word2Vec is a two-layer neural community. It takes text corpus as input and its output is a hard and fast of feature vectors for words in that corpus. For each word, we generate one hundred functions. These word vectors allow for numeric operations on words. For example, the vector of “king” delivered by way of the vector of “lady” could return the vector of “queen”. After walking Word2Vec, we set up a mapping from words to vectors.

A screenshot of a code editor with several tabs at the top: LDA.py, word2vec.py, applemod2.txt, main.py, LOF.py, and makefile.py. The active tab is word2vec.py. The code is as follows:

```
52
53 def initVocabModel():
54     logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
55     corporaFiles = os.listdir('./corpora')
56
57     generateVocabulary('./corpora/vocabulary.txt')
58
59     model = gensim.models.Word2Vec()
60     trainSentences = gensim.models.word2vec.LineSentence('./corpora/vocabulary.txt')
61     # print (trainSentences)
62     # input()
63     model.build_vocab(trainSentences)
64     model.train(trainSentences, total_examples=model.corpus_count, epochs=model.iter)
65     model.save('word2vecModel')
66
67 r = re.compile('[A-Za-z]+')
68 deleteSet = set(['a', 'an', 'and', 'or', 'the'])
69
70
```

Fig 4.2(i) Training Word2Vec Model

### 4.3 Phase 3- Clustering Phase

For a new test report, we first get the function vector of each sentence. We then use Agglomerative Clustering to segment the record. We first specify that we want to have N clusters. We then placed each sentence into its personal cluster, and iteratively merge the closest clusters till most effective N clusters continue to be. We then examine these clusters. If there exists a cluster that includes best one sentence, then we recognize that this sentence is a special one, as it's miles by no means merged into different clusters.

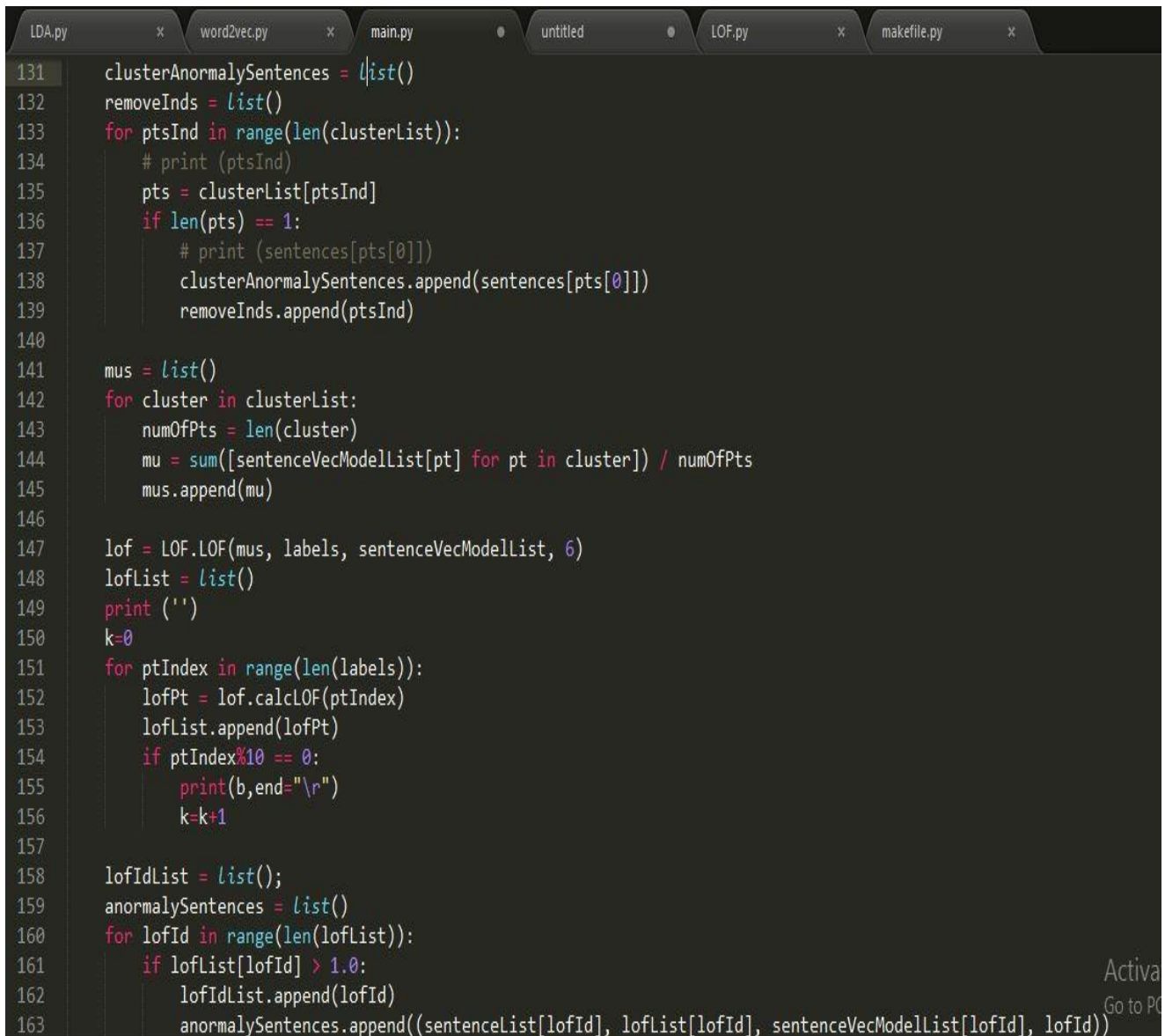
```
LDA.py x word2vec.py x main.py x LOF.py x makefile.py x
101
102 aggModellist = list()
103 silhouette_avg_list = list()
104 for n_cluster in range(2, 40):
105     aggModel = AgglomerativeClustering(n_clusters=n_cluster, linkage='average', affinity='cosine')
106     aggModel.fit(sentenceVecModellist)
107     aggModellist.append(aggModel)
108     silhouette_avg = silhouette_score(array(sentenceVecModellist), aggModel.labels_)
109     silhouette_avg_list.append(silhouette_avg)
110
111
112 n_clusters = 3
113
114 testAggModel = AgglomerativeClustering(n_clusters=n_clusters, linkage='average', affinity='cosine')
115 # print (sentenceVecModellist)
116 testAggModel.fit(sentenceVecModellist)
117 # print (testAggModel)
118
119 labels = testAggModel.labels_.tolist()
120
121 clusterList = [list() for _ in range(n_clusters)]
122
123 for senVecId in range(len(sentenceVecModellist)):
124     clusterList[labels[senVecId]].append(senVecId)
125
126 mycluster=clusterList[0]
127
128 # print (mycluster)
129 # input()
130
```

Fig. 4.3(i) Clustering the Test Document



#### 4.4 Phase 4- Anomaly Detection Phase

The nearby outlier component is based on a concept of a neighborhood density, wherein locality is given via nearest friends, whose distance is used to estimate the density. By comparing the neighborhood density of an object to the neighborhood densities of its pals, you can perceive regions of comparable density, and points that have a drastically lower density than their neighbors. These are considered to be outlier.



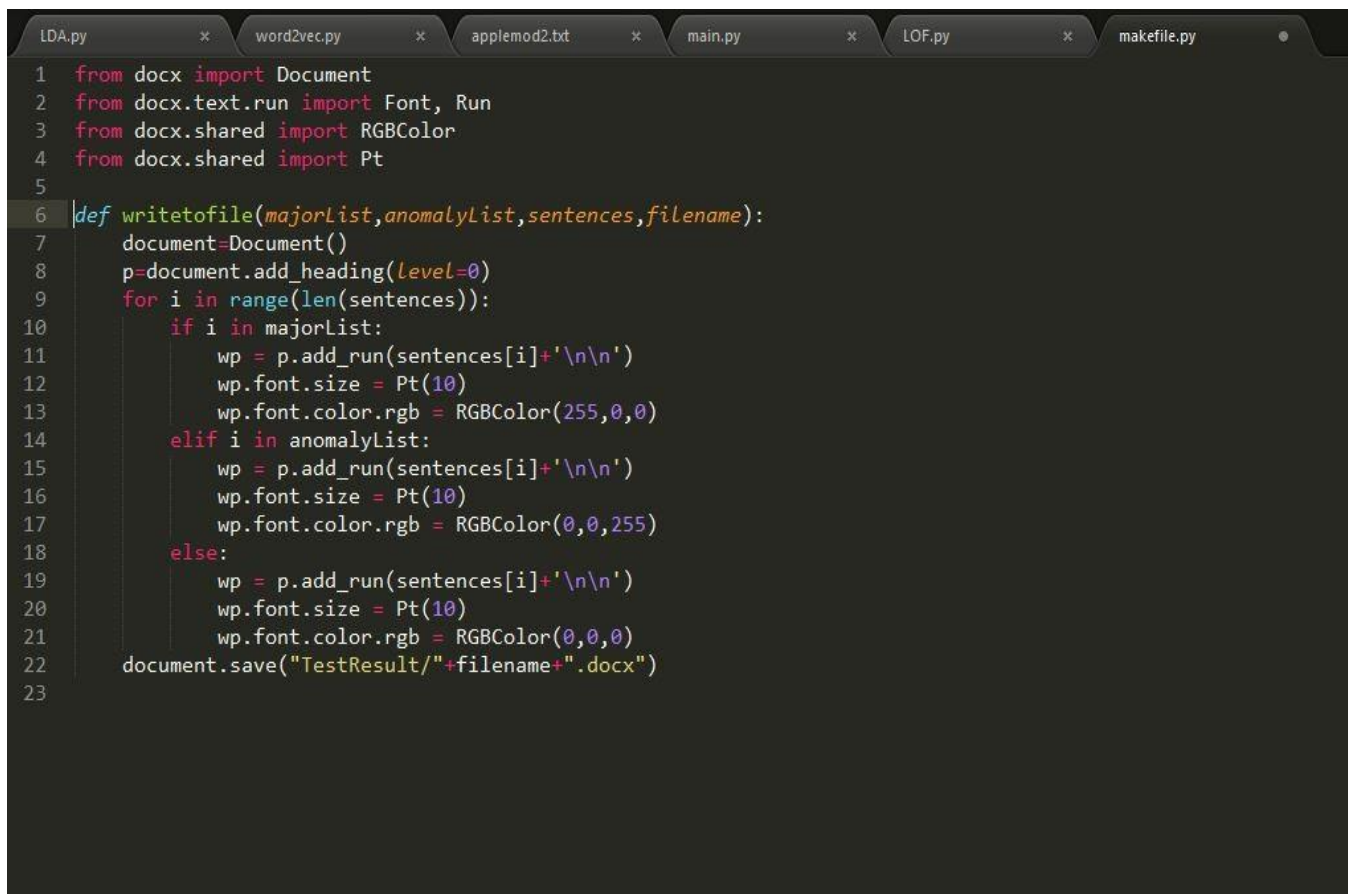
```
131 clusterAnomalySentences = list()
132 removeInds = list()
133 for ptsInd in range(len(clusterList)):
134     # print (ptsInd)
135     pts = clusterList[ptsInd]
136     if len(pts) == 1:
137         # print (sentences[pts[0]])
138         clusterAnomalySentences.append(sentences[pts[0]])
139         removeInds.append(ptsInd)
140
141 mus = list()
142 for cluster in clusterList:
143     numOfPts = len(cluster)
144     mu = sum([sentenceVecModellist[pt] for pt in cluster]) / numOfPts
145     mus.append(mu)
146
147 lof = LOF.LOF(mus, labels, sentenceVecModellist, 6)
148 lofList = list()
149 print ('')
150 k=0
151 for ptIndex in range(len(labels)):
152     lofPt = lof.calcLOF(ptIndex)
153     lofList.append(lofPt)
154     if ptIndex%10 == 0:
155         print(b,end="\r")
156         k=k+1
157
158 lofIdList = list();
159 anomalySentences = list()
160 for lofId in range(len(lofList)):
161     if lofList[lofId] > 1.0:
162         lofIdList.append(lofId)
163         anomalySentences.append((sentenceList[lofId], lofList[lofId], sentenceVecModellist[lofId], lofId))
```

Fig. 4.4(i) Anomaly Detection

## 4.5 Phase 5- Final Phase

In final phase we have to make a file in We have created **DOCX** file doing all the above task and then convert this **DOCX** file into **PDF** for the better presentation.

which important sentences are to be marked '**RED**', anomaly sentences are to be marked '**BLUE**', and other not important sentences not be marked, leave them as it is i.e. '**BLACK**'.



```
LDA.py x word2vec.py x applemod2.txt x main.py x LOF.py x makefile.py
1 from docx import Document
2 from docx.text.run import Font, Run
3 from docx.shared import RGBColor
4 from docx.shared import Pt
5
6 def writetofile(majorList, anomalyList, sentences, filename):
7     document=Document()
8     p=document.add_heading(level=0)
9     for i in range(len(sentences)):
10        if i in majorList:
11            wp = p.add_run(sentences[i]+'\\n\\n')
12            wp.font.size = Pt(10)
13            wp.font.color.rgb = RGBColor(255,0,0)
14        elif i in anomalyList:
15            wp = p.add_run(sentences[i]+'\\n\\n')
16            wp.font.size = Pt(10)
17            wp.font.color.rgb = RGBColor(0,0,255)
18        else:
19            wp = p.add_run(sentences[i]+'\\n\\n')
20            wp.font.size = Pt(10)
21            wp.font.color.rgb = RGBColor(0,0,0)
22        document.save("TestResult/"+filename+".docx")
23
```

Fig 4.5(i) DOCX file making

```
LDA.py x word2vec.py x main.py x LOF.py x makefile.py x
12 import subprocess
13 import time
14
15 def convert_to_pdf(filename):
16     filen='TestResult/'+filename+'.docx'
17     y=['-', '\\', '-', '/']
18     print ("")
19     for x in range (0,6):
20         b = "Making DOCX File " + '.'*x
21         print (b, end="\r")
22         time.sleep(0.5)
23     print ("")
24     print ("")
25     for x in range (0,6):
26         b = "Converting to PDF " + '.'*x
27         print (b, end="\r")
28         time.sleep(0.5)
29     print ("")
30     subprocess.call(['libreoffice', '--convert-to', 'pdf', '--outdir', 'TestResult/', filen])
31     # subprocess.call(['rm', filen])
32     print ('\n\nFile Saved With the Name', filename, '\b.docx in the TestResult Directory\n')
33     print ('\n\nFile Saved With the Name', filename, '\b.pdf in the TestResult Directory\n')
34
35 def Union(lst1, lst2):
36     final_list = list(set(lst1) | set(lst2))
37     return final_list
38
39 def Diff(li1, li2):
40     return (list(set(li1) - set(li2)))
41
42
```

Fig 4.5(ii) Converting to PDF

## CHAPTER 5

### TRAINING OUTPUT, TEST AND RESULT

#### 5.1 Phase 1 – Trained LDA Model Output

LDA model output showing top 30 words from each of the topic that are building the documents. In this the model is trained with top words that will be required to highlight the give text.

```
ariz@ariz-pc:/media/ariz/Local Disk/College & Training/College/Semesters/08th Sem/Major Project/Coding$ python loadLDA.py
Printing Top 30 Words from each Topic of LDA Model

TOPIC 1
0.037*"software" + 0.025*"customer" + 0.018*"may" + 0.018*"use" + 0.013*"s" + 0.012*"services" + 0.012*"agreement" + 0.011*
"license" + 0.008*"contractor" + 0.008*"terms" + 0.007*"2" + 0.007*"will" + 0.007*"information" + 0.006*"1" + 0.006*"comput
er" + 0.006*"party" + 0.006*"shall" + 0.006*"section" + 0.006*"16" + 0.006*"third"

TOPIC 2
0.022*"will" + 0.022*"tctl" + 0.011*"confidential" + 0.011*"retainer" + 0.011*"consultant" + 0.011*"shall" + 0.010*"informa
tion" + 0.008*"s" + 0.008*"company" + 0.008*"may" + 0.008*"per" + 0.006*"agency" + 0.006*"offer" + 0.006*"time" + 0.005*"pe
rformance" + 0.005*"rights" + 0.005*"training" + 0.004*"federal" + 0.004*"joining" + 0.004*"including"

TOPIC 3
0.025*"terms" + 0.023*"agreement" + 0.017*"will" + 0.015*"use" + 0.014*"products" + 0.014*"service" + 0.014*"program" + 0.0
12*"us" + 0.012*"information" + 0.010*"agree" + 0.010*"platform" + 0.010*"services" + 0.008*"may" + 0.007*"s" + 0.007*"api"
+ 0.007*"confidential" + 0.006*"release" + 0.006*"materials" + 0.006*"account" + 0.006*"pre"

TOPIC 4
0.019*"management" + 0.017*"support" + 0.017*"service" + 0.016*"business" + 0.014*"products" + 0.014*"learn" + 0.010*"help"
+ 0.010*"team" + 0.010*"partners" + 0.010*"project" + 0.010*"teams" + 0.010*"git" + 0.009*"self" + 0.009*"great" + 0.009*"
communication" + 0.009*"get" + 0.008*"code" + 0.007*"services" + 0.007*"provide" + 0.007*"customer"

TOPIC 5
0.019*"agreement" + 0.014*"may" + 0.014*"products" + 0.013*"party" + 0.013*"will" + 0.012*"use" + 0.009*"information" + 0.0
08*"s" + 0.008*"shall" + 0.008*"services" + 0.008*"notice" + 0.007*"order" + 0.007*"data" + 0.006*"including" + 0.006*"soft
ware" + 0.006*"user" + 0.005*"hosted" + 0.005*"terms" + 0.005*"applicable" + 0.005*"third"
```

Fig 5.1(i) Result of LDA trained model



## 5.2 Phase 2 – Trained Word2Vec Model Output

All word represented in a hundred-dimension vector layout, i.e. For each phrase there are 100 capabilities. These phrase vectors let in the numeric operation on words.

(i) Vector representation of word ‘apple’

```
ariz@ariz-pc:/media/ariz/Local Disk/College & Training/College/Semesters/08th Sem/Major Project/Coding$ python word2vecra
d.py
apple
[ 6.20687716e-02 -1.86354533e-01 -6.95587158e-01  4.83824313e-01
-2.55506426e-01 -3.20919156e-02  2.76965052e-02 -5.00236750e-01
-2.18342617e-01  1.29257023e+00  1.29905009e+00 -3.22487056e-01
-6.13210320e-01  1.06342101e+00 -1.20508503e-02  8.09740186e-01
 9.30909991e-01  1.11049032e+00  1.07623115e-01  1.58931684e+00
-1.71514714e+00  9.26415741e-01 -5.88379622e-01 -4.16073799e-01
-4.23576236e-01  2.46751055e-01  2.65782118e-01 -7.17319191e-01
-5.69699369e-02 -5.04516006e-01 -2.24108979e-01  2.20868824e-04
-6.20637611e-02 -6.89212799e-01  5.70509791e-01  9.73226130e-01
-3.41783129e-02 -4.00158048e-01  3.45828980e-01 -2.55101740e-01
 8.31853926e-01 -9.78629529e-01  4.45145994e-01  1.51514605e-01
-1.33554077e+00 -1.36320221e+00 -5.96417665e-01 -5.28007865e-01
 2.06239894e-01 -3.74080122e-01  6.62639976e-01 -1.45184159e-01
-9.52570215e-02 -7.44555593e-01  2.63388366e-01  1.07254899e+00
-1.33669829e+00  7.72973299e-01  9.64502990e-01  1.27908200e-01
-4.11107451e-01 -2.77583510e-01  5.14314055e-01  8.81555617e-01
 2.86406040e-01 -6.36091948e-01 -8.31589937e-01 -8.84314030e-02
-7.92171285e-02 -1.21844900e+00 -7.74188578e-01  4.71925020e-01
 2.31111512e-01 -4.69955444e-01 -1.05217230e+00  2.89409161e-01
 9.62719738e-01 -2.84995377e-01  7.75729537e-01  9.38036442e-01
-3.09667647e-01 -1.80315506e+00  7.07095325e-01 -4.25655693e-02
-1.16332281e+00 -2.67666429e-01  6.71458244e-02  1.02341437e+00
 8.32144260e-01  7.41804719e-01  9.33298707e-01  8.80131721e-01
-1.44576514e+00 -3.86254629e-04 -1.39473236e+00  2.44688421e-01
-1.56627089e-01 -1.26550531e+00 -5.12425721e-01 -1.07803023e+00]
```

Fig 5.2(i) Vector representation of word ‘apple’

(ii) Vector representation of word 'crime'

```
ariz@ariz-pc:/media/ariz/Local Disk/College & Training/College/Semesters/08th Sem/Major Project/Coding$ python word2vecra
d.py
crime
[ -1.11572839e-01 -1.23072706e-01  6.07641377e-02 -1.46414623e-01
 -2.34503541e-02  5.75567503e-03 -9.19763371e-02 -8.88661742e-02
  2.99133584e-02  1.12293735e-01  6.96648881e-02 -9.57809761e-02
  9.82298329e-02  2.06134811e-01  1.47650197e-01  8.16153809e-02
  1.09181218e-02 -5.98062165e-02 -6.50919899e-02 -5.96951768e-02
  7.37893060e-02 -2.16973603e-01 -1.01578943e-01  1.88667685e-01
 -1.59822652e-04 -2.29779273e-01 -2.11686179e-01 -3.37319560e-02
 -6.65012049e-03 -1.89996194e-02  6.87890798e-02 -8.27625860e-03
  1.17215790e-01  1.31185755e-01  4.50542569e-02  7.64913857e-03
  1.57965928e-01  5.73367067e-02  1.22834295e-01  1.05162285e-01
 -4.42530140e-02 -4.63910066e-02 -1.34173468e-01 -1.59915201e-02
 -3.15046385e-02 -6.49657771e-02 -6.22104714e-03 -8.45106840e-02
  1.85762063e-01 -1.13279432e-01 -1.82846002e-03 -2.97074988e-02
 -7.51509592e-02  4.45262752e-02 -1.75417289e-01  2.29322404e-01
  1.09622464e-01 -9.39051732e-02  2.18270551e-02  8.47785324e-02
 -4.22813743e-02  2.33022541e-01 -1.55433923e-01 -9.85899195e-02
  7.42919371e-03 -7.92248100e-02 -4.19525057e-02 -5.32367975e-02
 -2.44546816e-01  1.83112621e-01 -6.69097081e-02  4.69941944e-02
 -8.93437713e-02 -5.50960153e-02 -5.26644709e-03  1.05269834e-01
  2.80649383e-02  5.37814498e-02  2.63985604e-01 -3.93793732e-02
 -6.75470829e-02 -1.13722138e-01  7.94248357e-02  4.87115830e-02
 -1.07845850e-01 -1.49438471e-01 -5.36244474e-02 -4.73474823e-02
  4.45269085e-02  6.77485690e-02  8.05655029e-03  1.07562184e-01
 -9.78843719e-02 -9.42832008e-02 -7.30758952e-03 -1.79847702e-03
 -9.85860899e-02 -4.93746102e-02 -2.17408221e-02 -1.61019519e-01]
```

Fig 5.2(ii) Vector representation of word 'crime'



(iii) Vector representation of word 'agreement'

```
ariz@ariz-pc:/media/ariz/Local Disk/College & Training/College/Semesters/08th Sem/Major Project/Coding$ python word2vecra
d.py
agreement
[-0.77797771 -0.21342303 0.47116512 -0.63463455 0.26159191 -0.35629278
-1.09197354 -0.71748298 1.12291682 0.78544641 0.65450478 -0.54574996
-0.7863813 1.21609318 -0.00394175 0.23392418 0.56664467 1.12745798
-1.41502857 -0.24535741 0.71498513 0.22868253 -0.71453404 -0.42124331
-0.50901514 -0.43151906 -1.46814299 -0.49427661 0.090833 -1.19185972
-1.32399035 1.26043034 -0.36018988 -1.48716044 0.07004046 1.16881943
0.74869448 -1.94600391 0.20368038 -0.61196643 -0.99611348 -0.08119392
-2.05661225 -1.32704031 0.13455361 0.06870037 -1.27064621 0.29803514
-0.44134292 0.0780654 -0.57542336 -0.39331928 -0.41984734 0.89170647
-0.81650287 1.02153063 0.06072925 -0.43434337 0.3278546 0.3798362
0.29895955 0.57962149 -0.41665351 -0.03353633 0.60224086 -1.39344013
0.02209605 -0.5931527 0.17880669 -0.59018481 0.44980714 -0.04412804
0.0952402 -0.62367165 0.02174029 1.23828566 1.23611212 -0.55090123
1.33641553 -0.72382551 -0.57479477 -1.11712933 0.70213336 -0.65741289
-1.38272035 -0.55941045 -0.24460214 -0.85191023 -0.55226076 0.25692898
0.42165324 1.32254755 -0.34473991 0.79337412 -0.62741745 0.12905397
0.62525874 -2.2240603 -0.50204951 0.66607791]
```

Fig 5.2(iii) Vector representation of word 'agreement'

(iv) Vector representation of word 'license'

```
ariz@ariz-pc:/media/ariz/Local Disk/College & Training/College/Semesters/08th Sem/Major Project/Coding$ python word2vecra
d.py
license
[-0.26474124 0.29227811 -0.51677895 0.20405714 -0.57274091 0.15207016
-0.85859972 -0.65717179 0.40616083 1.51903391 0.86769754 0.11565617
-0.28007075 1.34615874 -0.14646652 0.19811687 0.81360304 1.32248354
-0.72417104 0.59546727 -0.10790403 1.15897894 -1.33263969 0.02167474
-0.1103171 -0.15447637 -1.07961667 -1.46780777 0.20316273 -1.26628399
-0.70782059 0.15282503 0.00620261 -1.2411375 -0.01994192 0.84513122
0.43809596 -0.39494494 1.33230162 -0.63704383 -0.62321615 -0.12341595
-0.78142595 0.00659894 -0.07520328 -1.24889159 -1.07683551 0.12899658
0.60815459 -0.18207939 -1.31855607 -0.19032399 -1.1081332 -0.75405318
0.00602797 0.7421723 -1.16128433 -0.28796583 -0.41333222 1.12159348
0.651986 0.64024884 0.51639348 0.76703078 -0.20310025 -1.47647512
-0.18955661 -0.55249697 -0.06831406 -0.45790637 -0.02114695 1.17570662
0.17404017 -1.08479965 -1.39181411 0.32865897 1.78857553 -1.01452112
1.20996463 -0.48592344 0.01126263 -1.90118217 0.88300425 -0.40832162
-1.86925423 -0.63286346 0.39293352 1.24432456 0.76558334 0.51862276
0.48889828 1.5760932 -0.93158895 0.25680533 -1.19732392 -0.68999779
0.77290136 -1.17055428 -1.59184003 0.75836432]
```

Fig 5.2(iv) Vector representation of word 'license'



(v) Vector representation of word 'liability'

```
ariz@ariz-pc:/media/ariz/Local Disk/College & Training/College/Semesters/08th Sem/Major Project/Coding$ python word2vecra
d.py
liability
[-0.98378742 -0.915371 0.87810576 -1.62294292 0.08432293 -0.36715171
-0.62350339 -1.54697609 1.45922136 0.25690073 1.14843881 -0.87284642
-1.04299653 1.57040441 -0.48342043 -0.89193738 0.32490376 0.63978136
-1.38604522 -1.30606496 0.91255802 -0.33985445 -0.08481397 0.36644989
-1.24426973 -0.90304399 0.09265305 -0.22277361 -0.7884686 -0.82280976
-0.55592698 0.14799438 -0.44888672 -0.50551867 -0.36431375 1.25394595
0.47059181 -0.95664817 -0.35997492 0.62838674 -0.4897379 -1.77638447
-0.56783897 -1.06652391 0.21265097 -0.14363888 -1.3575325 0.47589576
-0.92839587 -0.55409306 0.48778462 -0.06949104 0.22615665 1.32293022
-0.46579158 1.00814509 0.35001048 0.7193687 1.38361871 -0.45544106
-1.22264147 0.3750532 -0.68721682 0.08286062 0.60298467 -0.63192964
-0.58605927 0.14461038 0.22908482 0.02996208 -0.46407154 0.49319464
-0.30276075 -0.09857902 1.26599121 1.32837856 0.06788691 1.17116344
1.09220159 -0.09467357 -1.21421599 -0.04718962 0.11795384 -0.29060829
-0.67033547 -0.3429116 -0.48460901 -1.05697083 -0.09584218 1.03915274
0.41678587 -0.40842333 -0.16821936 0.34338194 -1.05317509 0.06376039
-0.35411912 -0.93415284 1.09755397 -0.91847688]
```

Fig 5.2(v) Vector representation of word 'liability'

(vi) Vector representation of word 'law'

```
ariz@ariz-pc:/media/ariz/Local Disk/College & Training/College/Semesters/08th Sem/Major Project/Coding$ python word2vecra
d.py
law
[-0.56558853  0.1189409  1.1019938 -0.13285221 -0.51347923  0.02465603
-1.60114753 -0.7523697  0.97182161  0.2737281  1.22056806 -0.50415915
-0.63248265  1.82023394  0.2807177  0.41212577  1.42300022  1.16313815
 0.26923004 -0.77325267  1.03858376 -1.17026973 -0.2302774  0.71556354
-1.13692451 -1.86046565 -0.06701101 -1.30613577  0.17562374 -0.89843363
 0.19133636  0.35224321 -0.44353449 -0.78715581 -0.13141824  1.10871232
-0.32301068 -1.04832375 -0.4122251 -0.71070176 -0.46021903 -1.21613252
-0.52450085 -1.29126155  0.2682699 -0.20361996 -0.84481841 -0.19710292
 1.13152277 -1.09517515 -0.43051255  0.08604269 -0.71945822  0.76395518
-0.17528032  1.34369445  0.6994096 -0.77272809  1.21424305  1.10796428
-0.14555916  1.71132708  0.30267543  0.05722215  0.01274284 -0.57502538
 0.70551801 -0.18024825 -1.60219407  0.57508034  0.33468962  0.75235808
-0.2189524 -0.26204205  0.48897526  1.47826505  0.5349741  0.56289226
 1.07261455  0.70993698 -1.22152698 -1.76608789  0.80394095  0.13439994
-0.31063873 -0.29787254 -0.57066154 -0.39512813  0.01356003  1.34905994
 0.70006269 -0.35902643 -0.23491347  0.81769145  0.46781707  0.34141389
 0.03169445 -1.66284657  0.27255034  0.25587061]
```

Fig 5.2(vi) Vector representation of word 'law'



(vii). Vector representation of word 'customer'

```
ariz@ariz-pc:/media/ariz/Local Disk/College & Training/College/Semesters/08th Sem/Major Project/Coding$ python word2vecra
d.py
customer
[-0.69410491 -0.40795898 -0.18186066 0.38072401 -0.22378358 0.46417713
 0.05053457 -0.26786986 -0.03806539 0.46839705 0.64981377 -0.54919243
-1.14085221 0.41417593 0.60056561 0.64826536 0.74645448 0.51602298
-0.5867334 0.29856169 -0.52103347 1.13389242 -0.98513138 -0.19090077
 0.04041405 -0.37711477 -0.74946547 -1.14029849 0.18714233 -1.18851936
-0.41216061 -0.19593176 -0.32588738 -1.36157024 0.16317818 0.16955577
-0.51908439 -0.31913173 0.00901913 -0.75264394 -0.02520345 -0.18460812
-0.86677909 0.55679148 -0.54673392 -0.99695909 -0.51313424 -0.4658969
 0.29757252 -0.36086151 -0.26242599 -0.31319734 -0.21307679 -0.55299973
-0.27758572 1.32889831 -0.29906908 0.66319376 0.3637335 0.18342248
 0.85414737 -0.20340213 1.21236539 0.62649423 0.01156487 -0.43799374
-1.05080342 -0.4978072 -0.30806619 0.1153285 0.27252463 0.25925732
 0.466297 -0.80428892 -0.59650415 0.16969246 0.75453413 -0.62734294
 1.11396527 0.01531559 -0.32802483 -1.132465 0.14986768 -0.41843367
-0.88111901 -0.20682602 -0.61022341 0.25479272 0.88312232 0.50918669
 0.14234105 1.00155509 -0.71331412 0.13324995 -1.38987124 -0.80008638
-0.11386164 -1.24227417 -0.6046558 0.18533532]
```

Fig 5.2(vii). Vector representation of word 'customer'

(viii). Vector representation of word 'software'

```
ariz@ariz-pc:/media/ariz/Local Disk/College & Training/College/Semesters/08th Sem/Major Project/Coding$ python word2vecra
d.py
software
[-0.42959058 -0.3893207 -0.97923994 0.0832792 -0.38225001 0.54133433
 0.22560114 -0.17894426 -0.12415802 1.55443704 0.84489036 -0.30181974
-0.40186328 0.57731348 0.5318715 0.54987407 0.86822915 1.11876488
-0.5481928 1.21072268 -1.47460938 0.84094214 -1.56845069 -0.17000756
-0.52142155 0.11593959 -0.71549118 -1.02562129 0.23005515 -1.17299104
-0.2140236 -0.5543254 0.12927112 -0.75442904 0.11371586 0.51207095
-0.12227812 -0.14680387 0.90060258 -0.12216337 0.75896072 -1.0540843
0.00752445 0.90937299 -0.51358461 -1.45748723 -0.61205786 -0.41499689
0.38072076 -0.13009572 -0.57582349 0.1130189 -0.18915631 -1.36291063
-0.07655398 1.25434697 -0.99184918 1.23538435 0.28950256 0.21421479
0.40095648 -0.17765735 0.78685808 1.05728936 0.12438108 -0.6260826
-1.626109 -0.71691281 -0.28778568 -0.46184897 -0.48152483 0.60181212
0.55520809 -1.15731192 -1.2500217 -0.38705674 1.33001709 -0.62818027
0.83392191 0.35585922 -0.19234468 -1.56983435 0.56053424 -0.09648629
-1.67940092 -0.1634025 0.10421289 1.25163829 1.45688725 0.51993763
0.6770817 1.34487009 -1.30717993 -0.12250895 -1.53651309 -0.89223236
0.03906918 -0.69483668 -1.16803944 -0.70086116]
```

Fig 5.2(viii). Vector representation of word 'software'

## 5.3 Phase 3 – Test And Result

### Test 1

(i) Test File One with No Irrelevant Sentence. All the words in this document need to read as not important highlighting is done in this.

80 Certain Services may display, include or make available content, data, information, applications or  
81  
82 materials from third parties ("Third Party Materials") or provide links to certain third party web sites. By  
83  
84 using the Services, You acknowledge and agree that the Application Provider is not responsible for  
85  
86 examining or evaluating the content, accuracy, completeness, timeliness, validity, copyright compliance,  
87  
88 legality, decency, quality or any other aspect of such Third Party Materials or web sites. The Application  
89  
90 Provider does not warrant or endorse and does not assume and will not have any liability or responsibility  
91  
92 to You or any other person for any third-party Services, Third Party Materials or web sites, or for any other  
93  
94 materials, products, or services of third parties. Third Party Materials and links to other web sites are  
95  
96 provided solely as a convenience to You. Financial information displayed by any Services is for general  
97  
98 informational purposes only and is not intended to be relied upon as investment advice.  
99  
100 Before executing any securities transaction based upon information obtained through the Services, You  
101  
102 should consult with a financial professional. Location data provided by any Services is for basic  
103  
104 navigational purposes only and is not intended to be relied upon in situations where precise location  
105  
106 information is needed or where erroneous, inaccurate or incomplete location data may lead to death.  
107

Activate Windows  
Go to PC settings to activate Windows.

**Fig 5.3(i)** Test File One with No Irrelevant Sentence



(ii) Result Corresponding to above Test File- here now the above text have been highlighted by the software marking the important words and separating them from the irrelevant words. Important sentences are to be marked '**RED**', anomaly sentences are to be marked '**BLUE**', and other not important sentences not be marked, leave them as it is i.e. '**BLACK**'.

certain services may display include make available content data information applications materials from third parties third party materials provide links to certain third party web sites

by using services you acknowledge agree that application provider is not responsible for examining evaluating content accuracy completeness timeliness validity copyright compliance legality decency quality any other aspect of such third party materials web sites

application provider does not warrant endorse does not assume will not have any liability responsibility to you any other person for any third party services third party materials web sites for any other materials products services of third parties

third party materials links to other web sites are provided solely as convenience to you

financial information displayed by any services is for general informational purposes only is not intended to be relied upon as investment advice

before executing any securities transaction based upon information obtained through services you should consult with financial professional

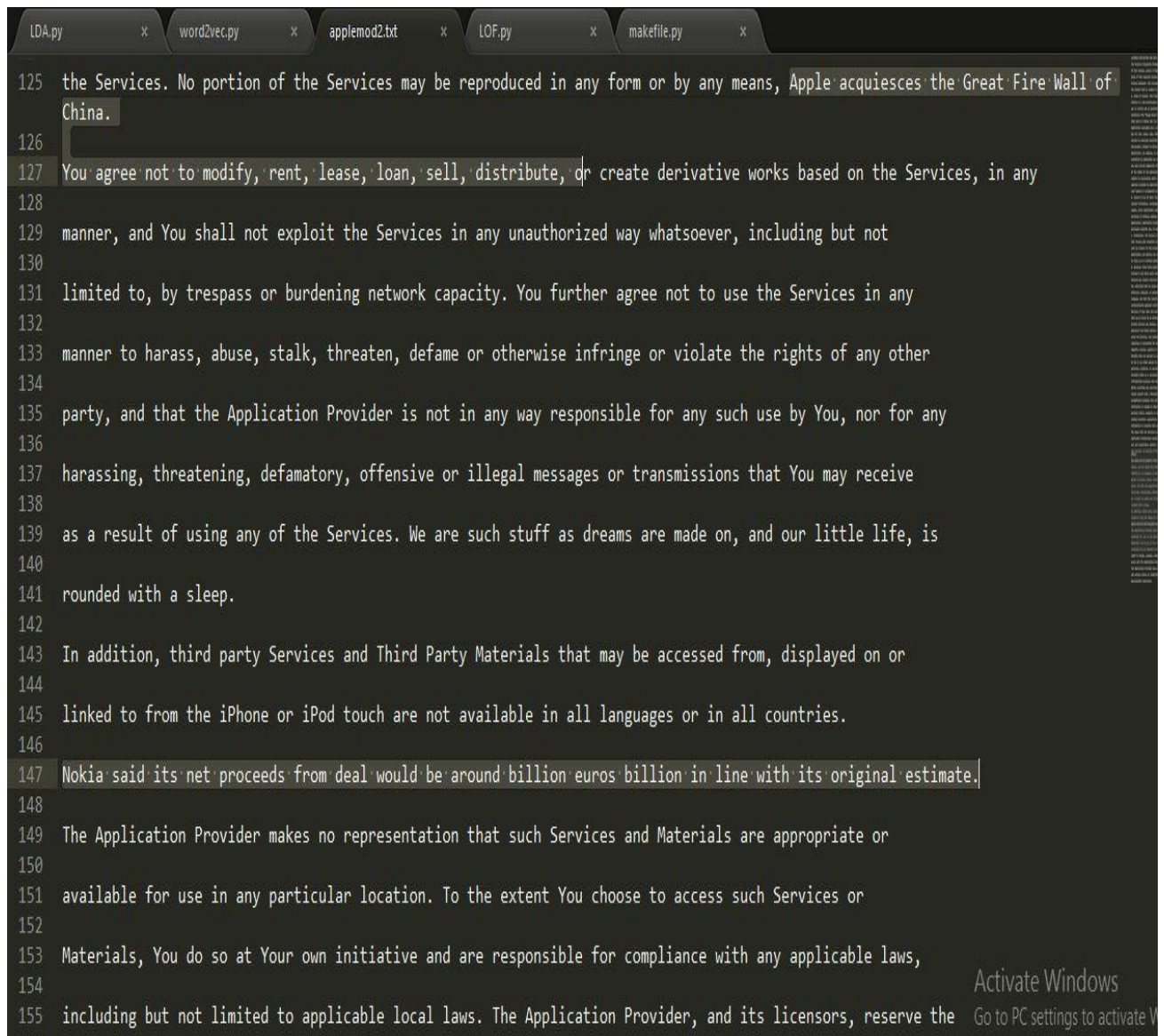
location data provided by any services is for basic navigational purposes only is not intended to be relied upon in situations where precise location information is needed where erroneous inaccurate incomplete location data may lead to death personal injury property environmental damage

activate Windows  
Go to PC settings to activate Wind

Fig 5.3(ii) Result Corresponding to above Test File

## Test 2

Test File two with No Irrelevant Sentence. All the words in this document need to read as not important highlighting is done in this.



```
LDA.py x word2vec.py x applemod2.txt x LOF.py x makefile.py x
125 the Services. No portion of the Services may be reproduced in any form or by any means, Apple acquiesces the Great Fire Wall of
    China.
126
127 You agree not to modify, rent, lease, loan, sell, distribute, or create derivative works based on the Services, in any
128
129 manner, and You shall not exploit the Services in any unauthorized way whatsoever, including but not
130
131 limited to, by trespass or burdening network capacity. You further agree not to use the Services in any
132
133 manner to harass, abuse, stalk, threaten, defame or otherwise infringe or violate the rights of any other
134
135 party, and that the Application Provider is not in any way responsible for any such use by You, nor for any
136
137 harassing, threatening, defamatory, offensive or illegal messages or transmissions that You may receive
138
139 as a result of using any of the Services. We are such stuff as dreams are made on, and our little life, is
140
141 rounded with a sleep.
142
143 In addition, third party Services and Third Party Materials that may be accessed from, displayed on or
144
145 linked to from the iPhone or iPod touch are not available in all languages or in all countries.
146
147 Nokia said its net proceeds from deal would be around billion euros billion in line with its original estimate.
148
149 The Application Provider makes no representation that such Services and Materials are appropriate or
150
151 available for use in any particular location. To the extent You choose to access such Services or
152
153 Materials, You do so at Your own initiative and are responsible for compliance with any applicable laws,
154
155 including but not limited to applicable local laws. The Application Provider, and its licensors, reserve the
156
```

Fig 5.3(iii) Test File Two with Irrelevant Sentences, Highlighted One

(ii) Result Corresponding to above Test File- here now the above text have been highlighted by the software marking the important words and separating them from the irrelevant words. Important sentences are to be marked '**RED**', anomaly sentences are to be marked '**BLUE**', and other not important sentences not be marked, leave them as it is i.e. '**BLACK**'.

materials in any way whatsoever except for permitted use of services

no portion of services may be reproduced in any form by any means apple  
acquiesces great fire wall of china

you agree not to modify rent lease loan sell distribute create derivative works  
based on services in any manner you shall not exploit services in any  
unauthorized way whatsoever including but not limited to by trespass burdening  
network capacity

you further agree not to use services in any manner to harass abuse stalk  
threaten defame otherwise infringe violate rights of any other party that  
application provider is not in any way responsible for any such use by you nor for  
any harassing threatening defamatory offensive illegal messages transmissions  
that you may receive as result of using any of services

we are such stuff as dreams are made on our little life is rounded with sleep

in addition third party services third party materials that may be accessed from  
displayed on linked to from iphone ipod touch are not available in all languages in  
all countries

nokia said its net proceeds from deal would be around billion euros billion in line  
with its original estimate

application provider makes no representation that such services materials are  
appropriate available for use in any particular location

Activate Windows  
Go to PC settings to activate Windows

Fig 5.3(iv) Result Corresponding to above Test File



## **CHAPTER 6**

### **CONCLUSION**

This project explains the process of text highlighting in lengthy legal documents with the help of the Python. The process of scraping documents using the EM for estimating LDA parameters, Agglomerative Clustering, Local Outlier Factor algorithms has also been briefly covered in the article. This project briefly describes the phases in which the project works. Also few sample test cases have been specified. I will recommend you to scrape any other document and see whether you can get a good highlighted summary of the document or not. . In order to save the readers time and effort this project will help the users to automatically highlight important parts of the document for a quick go through. At last this project successfully demonstrates the conversion of a legal document into highlighted text which proves useful to the users to save their time and effort to go through the entire document in a very less time and with much ease.

## CHAPTER 7

### FUTURE WORK

The methodology introduced here works entirely great, however isn't flawless. There are numerous enhancements which can be made by expanding the model multifaceted nature: Fast Thought Vectors, an on-going headway over the Skip-Thoughts approach can cause a huge decrease in preparing time and better execution. The skip-thought encoded portrayals have a dimensionality of 4800. These high dimensional vectors are not best reasonable for grouping purposes as a result of the Curse of Dimensionality. The dimensionality of the vectors can be diminished before bunching utilizing an Auto encoder or a LSTM-Auto encoder to grant further grouping data in the packed portrayals. Rather than utilizing extractive methodologies, abstractive outline can be executed via preparing a decoder arrange which can change over the encoded portrayals of the bunch focuses once more into regular language sentences. Such a decoder can be prepared by information which can be created by the skip-thought encoder. Be that as it may, cautious hyper-parameter tuning and engineering choices should be made for the decoder on the off chance that we need it to create conceivable and linguistically right sentences.

## CHAPTER 8 REFERENCES

1. Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." *the Journal of machine Learning research* 3 (2003): 993-1022.
2. Radim Rehurek, Optimizing word2vec in gensim. Nov 07, 2015.
3. <https://www.tensorflow.org/tutorials/word2vec>
4. <https://medium.com/jatana/unsupervised-text-summarization-using-sentence-embeddings-adb15ce83db1>
5. <https://www.analyticsvidhya.com/blog/2019/09/guide-automatic-highlight-generation-python-without-machine-learning/>
6. <https://monkeylearn.com/text-analysis/>