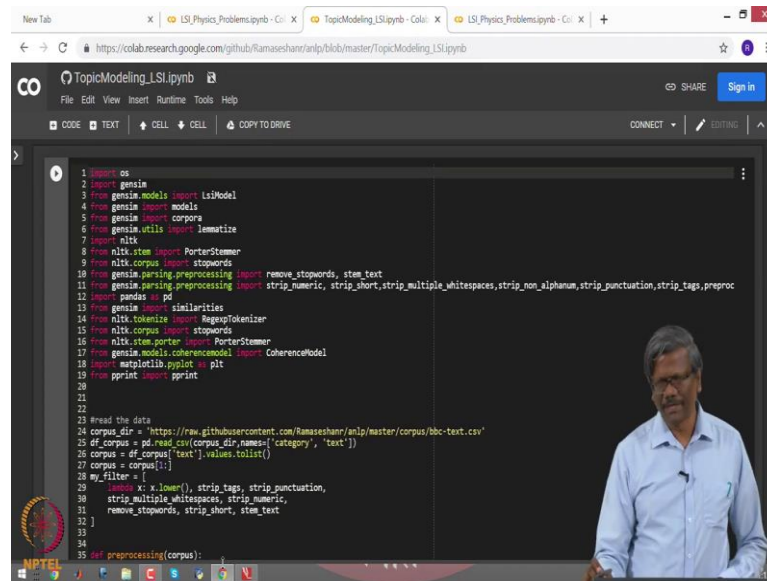


Applied Natural Language Processing
Prof. Ramaseshan Ramachandran
Visiting Professor at Chennai Mathematical Institute
Indian Institute Technology, Madras

Lecture - 19
Topic Modeling

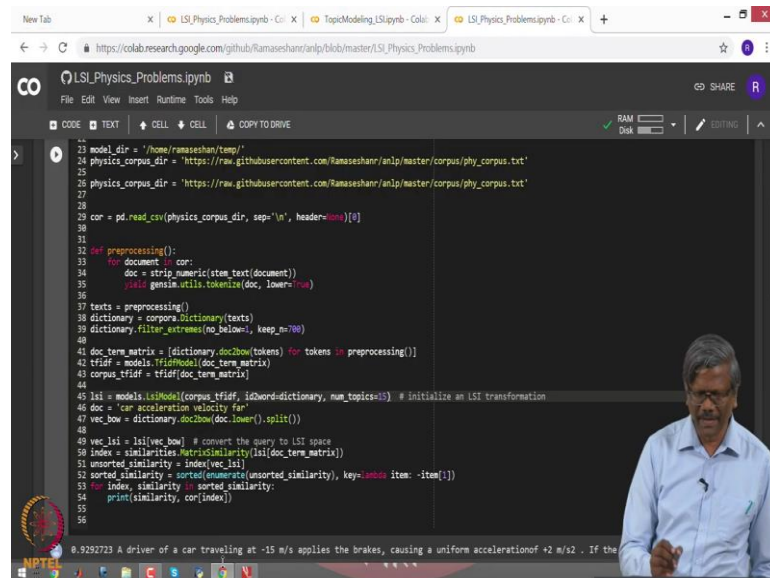
(Refer Slide Time: 00:15)



```
1 import os
2 import gensim
3 from gensim.models import LsiModel
4 from gensim import models
5 from gensim import corpora
6 from gensim.utils import lemmatize
7 import nltk
8 from nltk.stem import PorterStemmer
9 from nltk.corpus import stopwords
10 from gensim.parsing.preprocessing import remove_stopwords, stem_text
11 from gensim.parsing.preprocessing import strip_numeric, strip_short, strip_multiple_whitespaces, strip_non_alphanumeric, strip_punctuation, strip_tags, preprocess
12 import pandas as pd
13 from gensim import similarities
14 from nltk.tokenize import RegexpTokenizer
15 from nltk.corpus import stopwords
16 from nltk.stem.porter import PorterStemmer
17 from gensim.models.coherencemodel import CoherenceModel
18 import matplotlib.pyplot as plt
19 from pprint import pprint
20
21
22
23 #read the data
24 corpus_dir = "https://raw.githubusercontent.com/Ramaseshan/anlp/master/corpus/bbc-text.csv"
25 df_corpus = pd.read_csv(corpus_dir, names=['category', 'text'])
26 corpus = df_corpus['text'].values.tolist()
27 corpus = corpus[1:]
28 my_filter = []
29 for doc in corpus:
30     doc = doc.lower().strip_tags().strip_punctuation().strip_multiple_whitespaces().strip_numeric().strip_stopwords().strip_short().stem_text()
31
32
33
34
35 def preprocess(corpus):
```

So, here let us look at the topic modeling, but so one. So, earlier we saw the query related stuff right. So, in the transform domain, we are able to query a based on the keywords and related documents are retrieved using some similarities values. So, in this case, we are going to be looking at a different corpus. This corpus contains about five different topics. It contains topics related to business, entertainment, politics, sports, and technology ok. This is about five topics and the total document count is about 2225. And this corpus contains about 27,750 unique tokens ok.

(Refer Slide Time: 01:12)



```
23 model_dir = '/home/ramaseshan/temp/'
24 physics_corpus_dir = 'https://raw.githubusercontent.com/Ramaseshan/anlp/blob/master/LSI_Physics_Problems.ipynb'
25 physics_corpus_dir = 'https://raw.githubusercontent.com/Ramaseshan/anlp/master/corpus/phy_corpus.txt'
26
27
28 cor = pd.read_csv(physics_corpus_dir, sep='\\n', header=None)[0]
29
30
31
32 def preprocessing():
33     for document in cor:
34         doc = strip_numeric(strip_text(document))
35         # Use gensim utils.tokenize to tokenize the document
36         # Use gensim utils.tokenize to tokenize the document
37 texts = preprocessing()
38 dictionary = corpora.Dictionary(texts)
39 dictionary.filter_extremes(no_below=1, keep_n=700)
40
41 doc_term_matrix = [dictionary.doc2bow(tokens) for tokens in preprocessing()]
42 tfidf = models.TfidfModel(doc_term_matrix)
43 corpus_tfidf = tfidf[doc_term_matrix]
44
45 lsi = models.LsiModel(corpus_tfidf, idf_dictionary, num_topics=15) # Initialize an LSI transformation
46 doc = 'car acceleration velocity far'
47 vec_bow = dictionary.doc2bow(doc.lower().split())
48
49 vec_lsi = lsi[vec_bow] # convert the query to LSI space
50 index = similarities.MatrixSimilarity(lsi[doc_term_matrix])
51 unsorted_similarity = index(vec_lsi)
52 sorted_similarity = sorted(enumerate(unsorted_similarity), key=lambda item: -item[1])
53 # index, similarity
54 print(similarity, cor[index])
55
56
```

0.9292723 A driver of a car traveling at -15 m/s applies the brakes, causing a uniform acceleration of +2 m/s². If the

The normal setup especially in the case of the junction, if you do not provide any values as part of the model, will take the k value as 200. So, in this case I am just taking the k value as 200. And then I am going to be keeping 25000 words. So, our matrix size is going to be 25000 by 200 ok, so that is your original term-document matrix.

Again in this case I am computing the tfidf, and I have added little more additional functions for the pre-processing. One is I am taking off all the numeric using this strip numeric function and removing stop words given by the gensim, you may also want to add your own stop words to that. And then I am going to strip words which are smaller than 3, or smaller than or equal to 3.

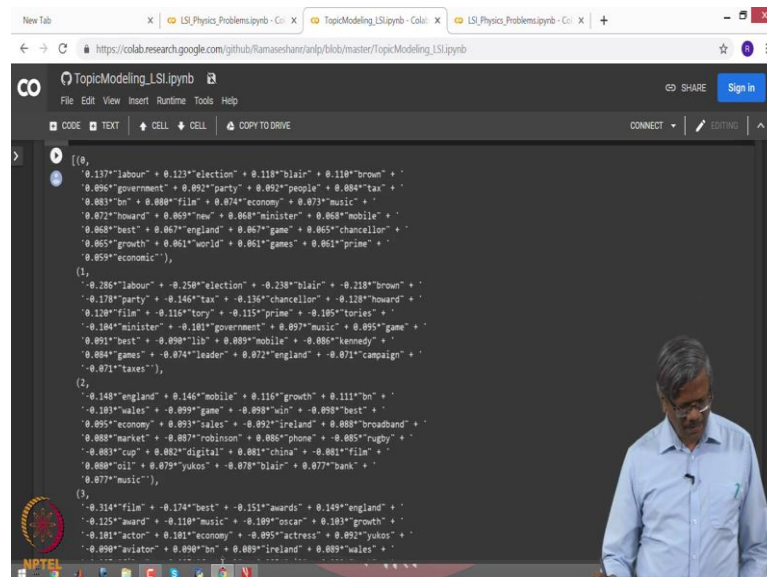
And I am not stemming it, I am removing the punctuations. If there is any tag that is available as part of the document that is also being removed. So, the pre-processing does all this task and then returns set of tokens ok. It is very similar to what we saw earlier is ok. So, we have done the pre-processing. Again for your convenience, I have kept the corpus which you can pick up from this particular location ok.

So, again I am keeping 25000 words, and I am not using this standard one that we used earlier. So, there we use the number of topics as 15 right in the previous problem. So, here I am not specifying anything we can specify that as well I can just leave it as if to the default value. I am going to see how it is defining this topic; the idea in this particular demo is to find out the number of topics in the given corpus.

So, nowhere we have specified the number of topics except in the last one where we are restricting the number of topics that it wants to this less right. So, once we have set up the text and then created the model. So, we can print the topics where we can pick up how many words you want in terms of describing the topic.

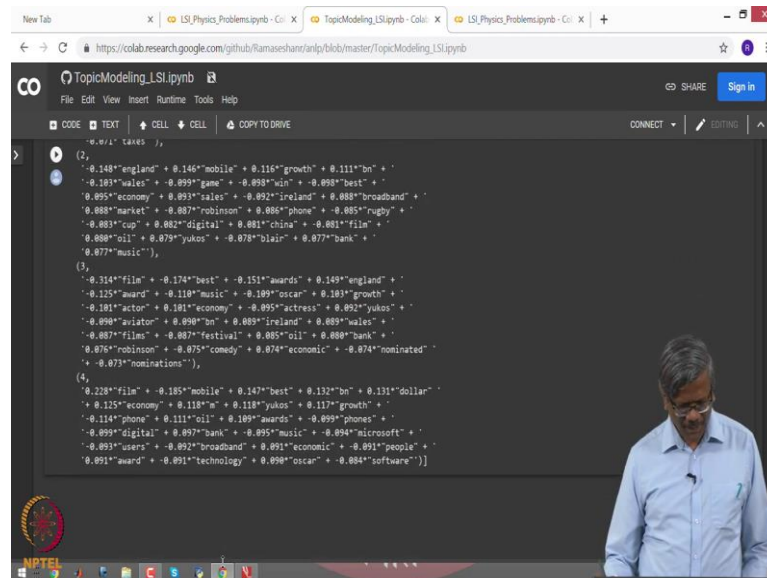
So, then that is the variable that you can change. So, here there is a variable that we can change or here the number of tokens or the number of topics that we can change. The number of topics here refers to the k in the Eigen matrix ok. And then we are going to be printing only five topics, and each topic will contain 25 words. So, I have computed this earlier for you.

(Refer Slide Time: 04:46)



```
[[0,
 0.137*"labour" + 0.123*"election" + 0.118*"blair" + 0.118*"brown" +
 0.896*"government" + 0.892*"party" + 0.892*"people" + 0.884*"tax" +
 0.883*"bn" + 0.888*"film" + 0.874*"economy" + 0.873*"music" +
 0.872*"howard" + 0.869*"new" + 0.868*"minister" + 0.868*"mobile" +
 0.868*"best" + 0.867*"england" + 0.867*"game" + 0.865*"chancellor" +
 0.865*"growth" + 0.861*"world" + 0.861*"games" + 0.861*"prime" +
 0.859*"economic"),
(1,
-0.786*"labour" + -0.258*"election" + -0.238*"blair" + -0.218*"brown" +
-0.178*"party" + -0.146*"tax" + -0.136*"chancellor" + -0.128*"howard" +
 0.128*"film" + -0.116*"tory" + -0.115*"prime" + -0.105*"tories" +
-0.104*"minister" + -0.101*"government" + 0.097*"music" + 0.095*"game" +
 0.091*"best" + -0.090*"11b" + 0.089*"mobile" + -0.086*"kennedy" +
 0.084*"games" + -0.074*"leader" + 0.072*"england" + -0.071*"campaign" +
-0.071*"taxes"),
(2,
-0.148*"england" + 0.146*"mobile" + 0.116*"growth" + 0.111*"bn" +
-0.103*"wales" + -0.099*"game" + -0.098*"win" + -0.098*"best" +
 0.095*"economy" + 0.093*"sales" + -0.092*"ireland" + 0.088*"broadband" +
 0.088*"market" + -0.087*"robinson" + 0.086*"phone" + -0.085*"rugby" +
-0.083*"top" + 0.082*"digital" + 0.081*"china" + -0.081*"film" +
 0.080*"oil" + 0.079*"yukos" + 0.078*"blair" + 0.077*"bank" +
 0.077*"music"),
(3,
-0.314*"film" + -0.174*"best" + -0.151*"awards" + 0.149*"england" +
-0.125*"award" + -0.118*"music" + -0.109*"oscar" + 0.103*"growth" +
-0.101*"actor" + 0.101*"economy" + -0.095*"actress" + 0.092*"yukos" +
-0.090*"aviator" + 0.090*"bn" + 0.089*"ireland" + 0.089*"wales" +
```

(Refer Slide Time: 04:49)



```
(2,
 -0.148*england + 0.146*mobile + 0.116*growth + 0.111*bn +
 -0.183*wales + -0.099*game + -0.098*win + -0.098*best +
 0.095*economy + 0.093*sales + -0.092*ireland + 0.088*broadband +
 0.088*market + 0.087*robinson + 0.086*phone + 0.085*rugby +
 -0.083*cup + 0.082*digital + 0.081*china + -0.081*film +
 0.080*oil + 0.079*yukos + -0.078*blair + 0.077*bank +
 0.077*music),
(3,
 -0.314*film + -0.174*best + -0.151*awards + 0.149*england +
 -0.125*award + -0.118*music + -0.109*oscar + 0.103*growth +
 -0.101*actor + 0.101*economy + 0.095*actress + 0.092*yukos +
 -0.090*aviator + 0.089*ireland + 0.089*wales +
 -0.087*film + 0.087*festival + 0.085*oil + 0.080*bank +
 0.076*robinson + 0.075*comedy + 0.074*economic + -0.074*nominated +
 -0.073*nominations),
(4,
 0.228*film + -0.185*mobile + 0.147*best + 0.132*bn + 0.131*dollar +
 0.125*economy + 0.118*bn + 0.118*yukos + 0.117*growth +
 -0.114*phone + 0.111*oil + 0.109*awards + -0.099*phones +
 -0.099*digital + 0.097*bank + -0.095*music + -0.094*microsoft +
 -0.093*users + -0.092*broadband + 0.091*economic + -0.091*people +
 0.091*award + -0.091*technology + 0.090*oscar + -0.084*software)]
```

And we can find out there are five topics printed by this application. The first one corresponds to let us find out to what. So, it has labor, election, Tony Blair, I think Brown government, party, people, tax, film, economy, music, new minister, mobile, game, chancellor, growth, world, games and so on right.

So, it contains lots of words that you will find in politics as well as in the entertainment and music ok. Let us see what it has done to the second one. So, we have labor, election, Blair, you see the values being small right in the negative side. So, again it is not very clear, it has not picked up the class very clearly. Let us look at the third one. There is a growth, billion, sales, economy, Ireland, broadband, market, phone, rugby, cup, digital, you can see even this sports-related to thinking our being mixed up in this right.

And then let us look at the fourth one, so, we are not very successful in terms of finding out the right class. Let us look at the last one film, the best award, England, award, music, Oscar, growth, actor, economy, actress, aviator, Ireland, Wales, films, festival, Robinson, nominated, nominations and so on. So, we can see a lot of terms related to entertainment.

Let us look at the last one. We have mobile, billion, dollar, economy, phone, digital, bank, Microsoft, users, broadband, economic, people, award, technology, Oscar, and software. Again it is there is a mix of both entertainment and technology in this.

So, how do we fine-tune this? So, the way you want to do is I am going to leave this again as an exercise that will also help you learn this library, but keep changing these values here, and then here the number of topics, the number of words. So, let us see if I can reduce the number of words to let say 10, and see what happens ok. So, I have to use the same one, I think there was some issue with respect to logging in. So, let me again go through them or is it similar? Yeah, it is the same, yeah, it is the same as what we saw ok. Let us change the values instead of 15, 25, let us make it 10.

So, this somewhat better you know because the number of terms is less. So, here in the first one, we see labor, election, Blair, Brown, government, party, people, tax and film. I think most of what we see is related to election or politics right. So, let us see the second one, labor, election, Blair, Brown, party, tax, chancellor, Howard, film and Tory. This looks like more than what we saw there again right.

The third one we see, again the first not very clear in terms of identifying which class it really belongs to look at the third one. We have we just look at only be positive once in terms of the weights England, Wales, game, win, best right. So, game and win, England Wales maybe sounding like sports.

Let us look at the fourth one film, best awards, award, music, Oscar, actor, economy right. So, if you look at the positive ones, we have film, best awards, award, music, Oscar, actor. So, it could be related to entertainment. Let us look at the last one film, best, dollar, economy, yukos, growth, phone, maybe it is both business and some technical aspect into its. By reducing the number right, we can see the word that is contributing to that. By keep changing the values here right instead of keeping 25000, keep only about 10,000 thousand and then see how it does.

So, by fine-tuning this you will be able to get to some optimal class for the given data that you have provided. So, it is not perfect right. It is not really a good classifier you know it is not able to really do a great job in terms of doing the classification because of a lot of noise that you find in the text. The noise comes because of the large number of vocabulary that we have, and many of them only have about very small contribution with respect to the frequency or tfidf, so that could be one of the biggest problems the reasons tends from the fact that and when you look at the sigma values ok.

(Refer Slide Time: 11:50)

The slide is titled "DEMO" and features a speaker on the left. The main content is a diagram with handwritten red annotations. The diagram consists of several interconnected boxes and arrows. The handwritten notes include: "1. Query Processing", "2. Topic Modeling", "1/2 27750", "n = 309421", "25000 x 200", "sigma_1 = 6.23", "sigma_2 = 1.3", and "100". There are also some arrows and boxes in the diagram.

Let me go back to, so the sigma values if you look at this for the BBC text sigma 1 equal to 6.23. And then if you look at this sigma 200 is equal to 1.3 see that is the problem correct. So, if you had seen in the image demonstration, the sigma values of the last one are very small, very negligible. Here there is a contribution. The contribution comes because of a lot of vocabulary words, a lot of words in the vocabulary that do not have a lot of frequencies that is why you know we are not able to really find the pattern. And if you have more number of pattern, the values here in the sigma would be very small ok.

So, we can say that because of the noise factor involved in the document, we are not able to really classified in a clean fashion. So, how can we do that, how can we do this job better, so that is where we have to go and then lookout the corpus, and then find out what kinds of a word that are contributing the each one of those maybe even in the case of the classification right what they have provided? They probably would have classes is there all manually classified in there could be some error there too is not it?

And then if you look at the sports and technology part, or business and politics, there could be a lot of words that are available as common words. So, that could also be one of the reasons why it is not really doing a great job. Another parameter that I have noted is it contains about 30,000 nouns. So, it contains 27,750 unique words. And then if you look at the noun count, it is about 30,000, not 30,300, and 9421 OKs, so that is the total count.

So, what we have done if we have taken about 25000 into 200 as our matrix size. And still, because the contribution of this sigma all throughout is really corrupting our application. So, how do we get this reduced, only if you are able to capture a lot of patterns that are represented by the first one right, so you will be able to do a reasonably good job? For us to do this is job is to make sure that we are trying to reduce this, and the contribution of this sigma is very small when it goes beyond the point.

For example, if you are able to cut its tool let say 100, the contribution from 100 to 200 to be very small and insignificant, whereas in this case it is not a joke. And also it the sigma value reduces very slowly if you have this as 200 and this is 6 right. So, it is like this should be the case. So, it should be like quickly, it should dk down ok. So, what I you suggest is taking a close look at many of those documents and then see what we can do to improve this, so that is the first step. So, you need to really look at that and then see what kind of pre-processing that we can do.

So, looking at the corpus closely is always about finding out how to do the pre-processing and eliminate certain noise is the thing that is not really going to contribute to our process. And then the second one is to look at the parameters that you have that I have shown, and then see and then try to fine-tune them one by one, and then see how they really contribute to the application also. So, with this, we conclude this session on the word to vectors.