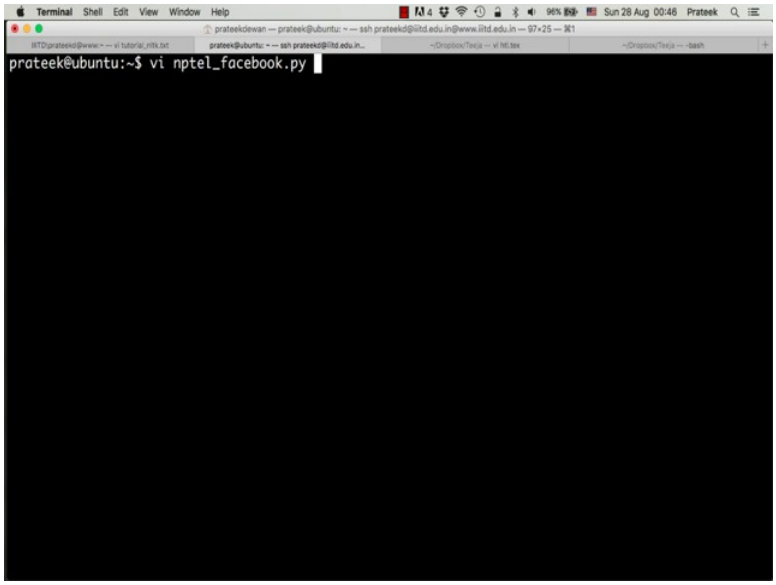
Privacy and Security in Online Social Networks
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 25
Tutorial 5: Analyzing text using Python NLTK

Hi everyone, welcome to another tutorial for the course Privacy and Security in Online Social Media. In the last few tutorials, we have seen how to collect data from Facebook and Twitter and we have seen some basics of python and Linux - Ubuntu and we have also seen some parts of social network analysis using Gephi.

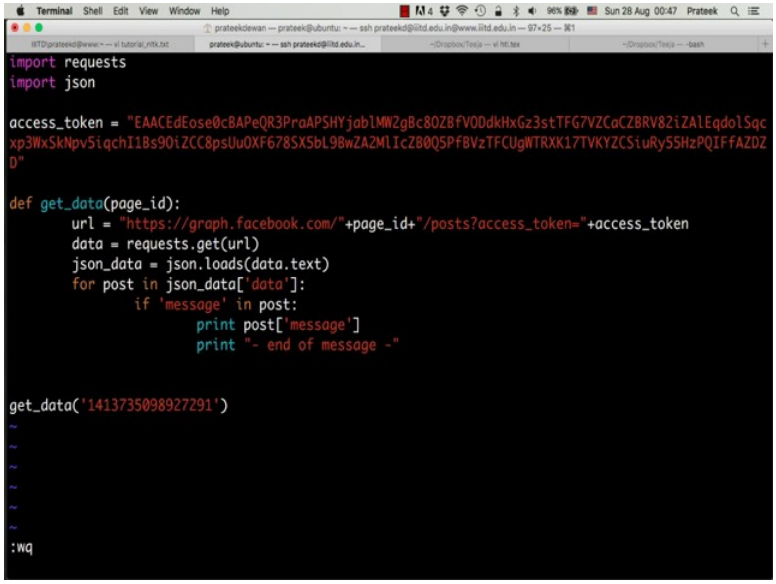
In today's tutorial, what we will learn is how to process the data that you have collected from Facebook and Twitter. Specifically, today we will be looking at the basics of working with textual data that you have obtained.

(Refer Slide Time: 00:50)



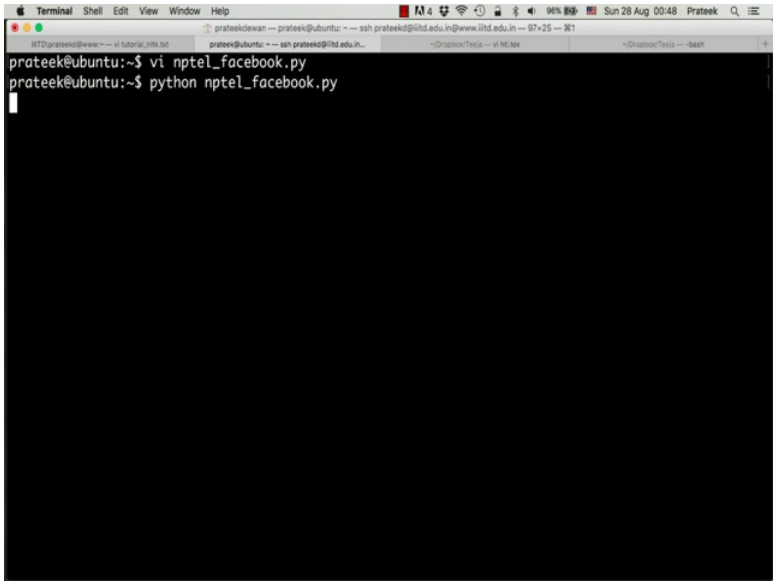
Now, let us go back to the Facebook data collection script.

(Refer Slide Time: 00:54)



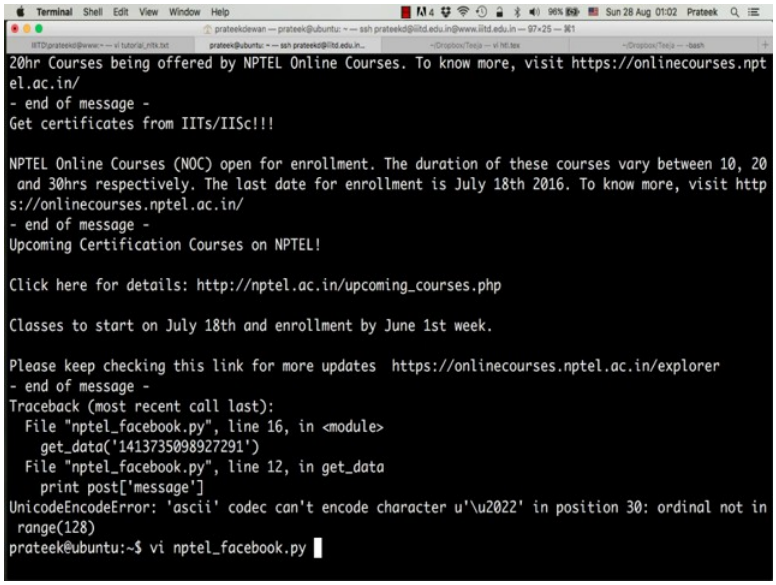
And if you remember, this is what we did to collect data from Facebook, we created a simple function. We defined a URL which was pointing to the graph API and we used a requests library to extract data from Facebook using this URL and if you remember, we saw how to print the message field present in Facebook posts. So, let us run this code again just to revise what we did in the first couple of weeks.

(Refer Slide Time: 01:23)



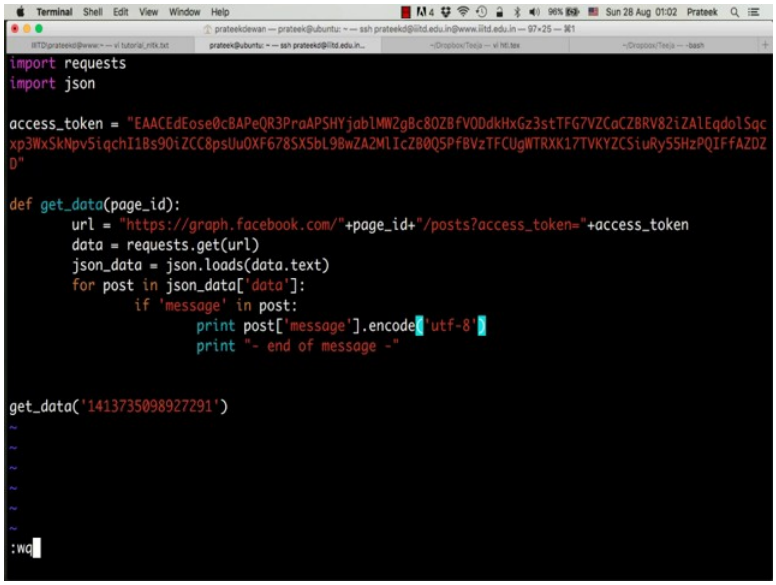
So, we say python space nptel underscore facebook.

(Refer Slide Time: 01:30)



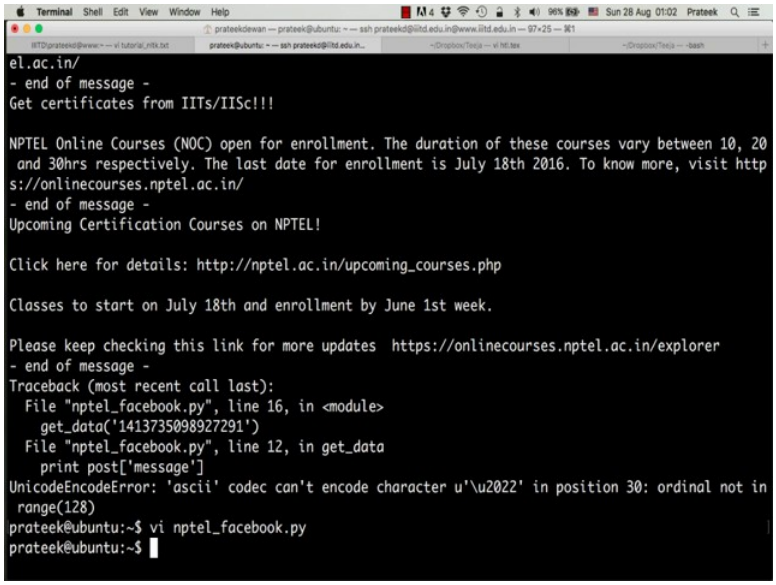
And it will print the message part of the posts that the NPTEL page has done. So, there is an error here. So, what happens is by default, when you are printing something on your terminal, if the data is not in the standard ASCII format and if there is a Unicode data; by default python uses ASCII to print character. Now, if it encounters a Unicode character it is not able to print it and it usually throws an error. So, if you read this error it says that ASCII codec cannot encode character u and some code in position 30. So, any string which starts with a u in the beginning is usually a Unicode string. Now, to handle such strings what we can do is.

(Refer Slide Time: 02:17).



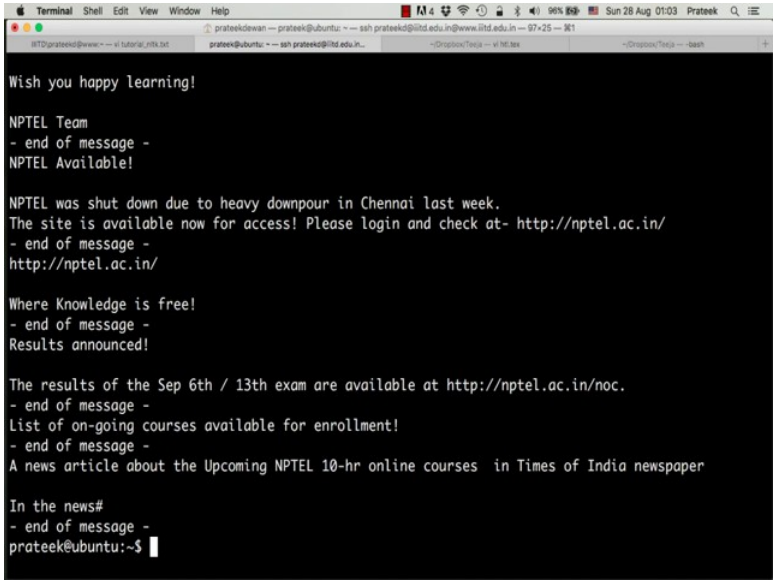
We can just add dot encode UTF-8 in the script. So, what this will do is, this will encode all the messages or all the text that we are getting into UTF-8 format and then print it. So, if there is any Unicode character in there which is not the standard a to z or 0 to 9 kind of a character, this will convert it to the whatever character it is, for example, Hindi text for example, Arabic text; those are text which are not contained in the ASCII character set. So, they can be printed using UTF-8 encoding. Now, once you have done this.

(Refer Slide Time: 03:04)



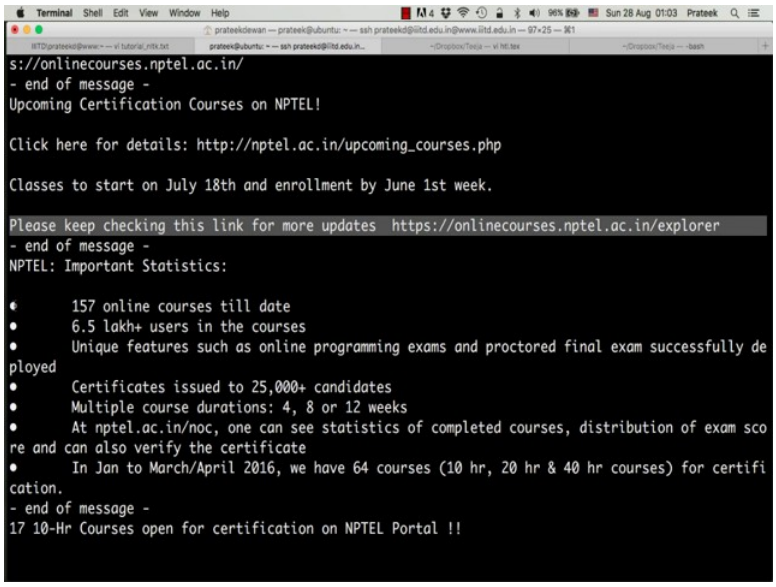
Now, let us try to run this script again.

(Refer Slide Time: 03:06)



And there you go, there is no error.

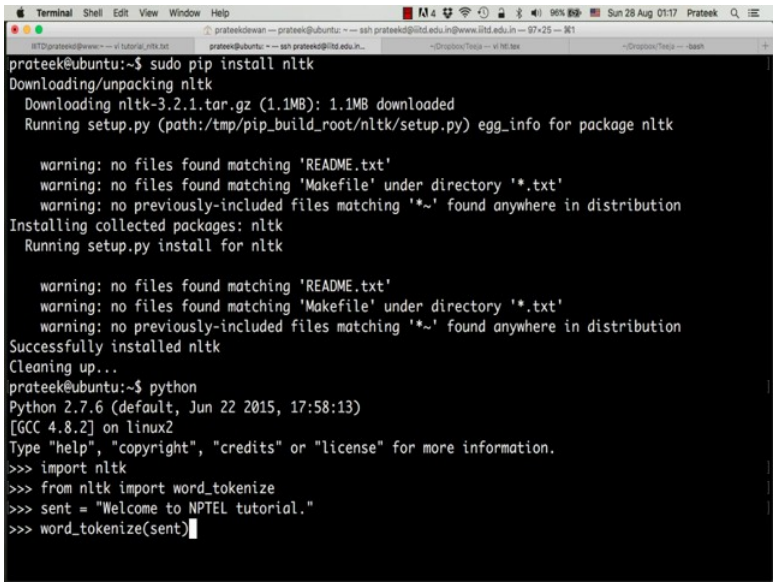
(Refer Slide Time: 03:12)



And if you scroll back, we can probably look at what character it was. So, this bullet point here this is not a standard ASCII character and this is where the error was coming. If you remember, this was the last post that we received and after that there was an error message. This is probably what caused the error. Now to start processing text, what you need is a library which can handle text. So, in python there is a library which is called natural language tool kit or NLTK in short. Let us install that first. So, you say sudo pip

install NLTK.

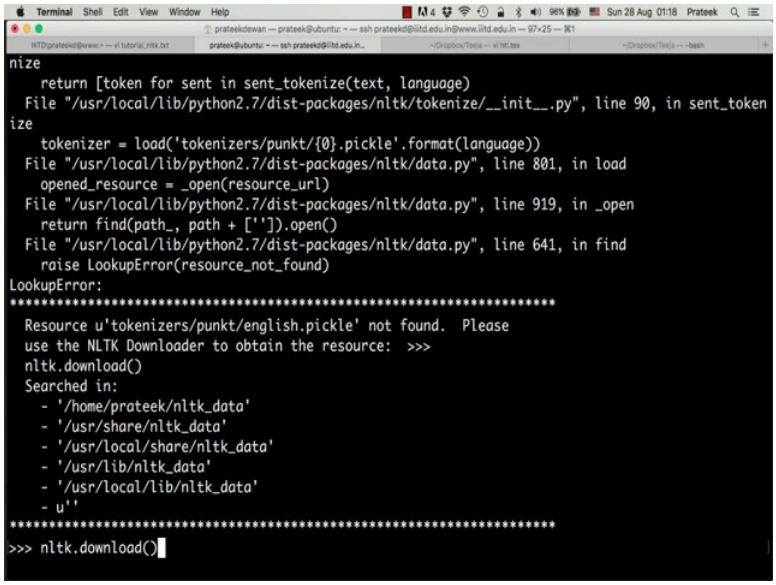
(Refer Slide Time: 03:47)



And press enter. Now, nltk is installed, if you go to the python shell and say import nltk, the import is successful, but interestingly just downloading and installing nltk does not suffice to perform the basic functions that you need.

So, for example, if you want to convert a sentence into a set of words, there is something called word-tokenizer. So, you can import it from nltk by saying, from nltk import word underscore tokenize and let us create a string say a sentence is equal to 'Welcome to NPTEL tutorial', now what this word underscore tokenize is supposed to do is when we give it a string. It is supposed to give us a list of the words present in the sentence. So, if we say word underscore tokenize sent.

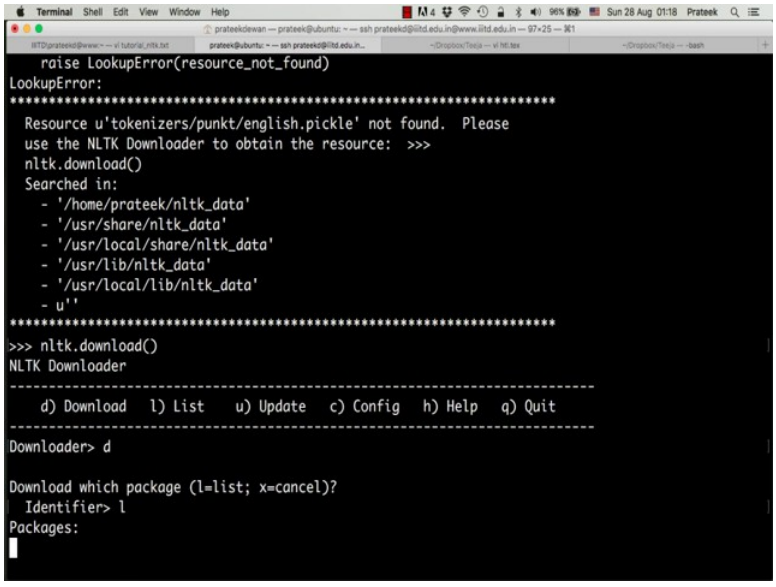
(Refer Slide Time: 04:59)



It is supposed to give us the words in the sentence, but it throws an error instead, now why is that that is because the nltk library requires some data to perform the basic operations, that data is present in a set of corpuses.

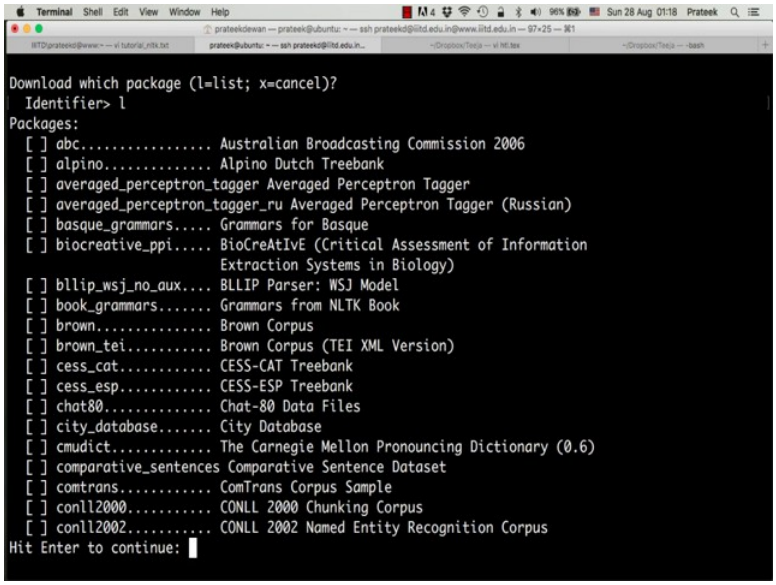
So, if you look at the error message carefully, it says that you should use this nltk dot download function to download the resources that are required to perform this operation. So, you just type nltk dot download and press enter and you will see this nltk downloader menu appear now type d and press enter.

(Refer Slide Time: 05:32)



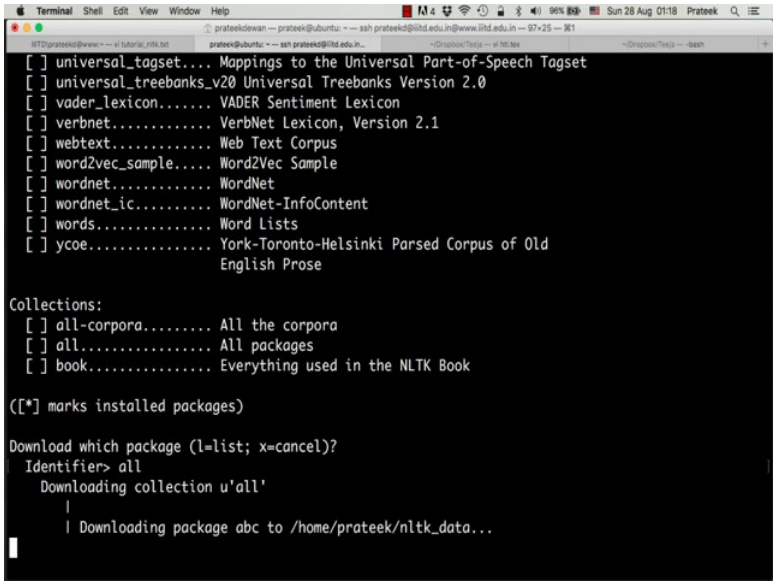
So, the downloader will ask you which package you want to download, now type l and press enter to see the list of available packages.

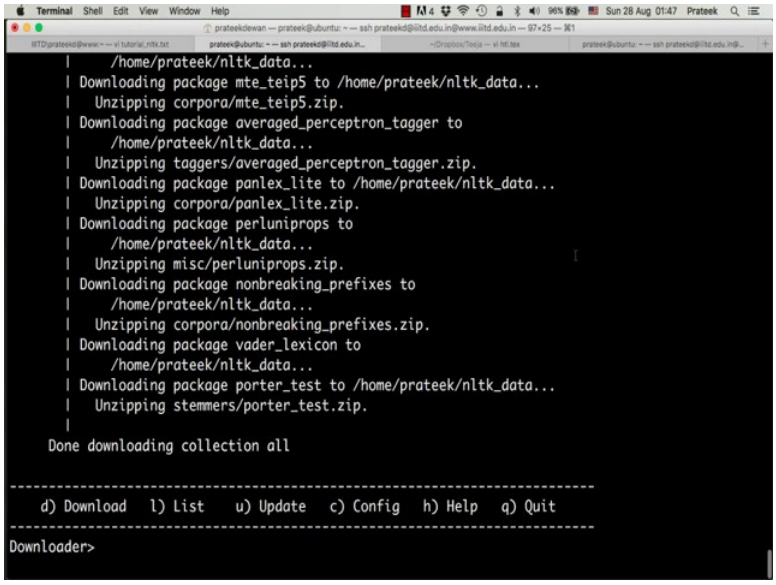
(Refer Slide Time: 05:42)



Now, there is a whole big list of packages that can be used with the NLTK library. So, instead of selecting the ones we want, what we will do is, we will just type all a double l and press enter which will allow us to download all the packages which are available.

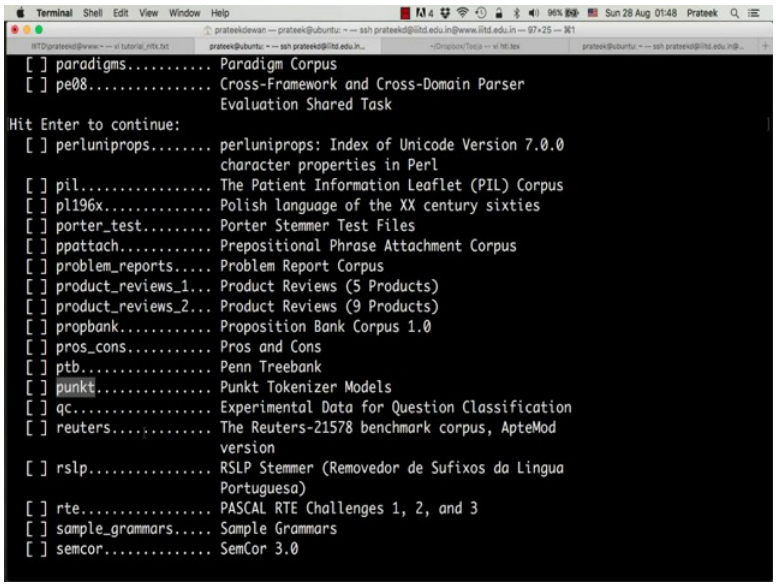
(Refer Slide Time: 05:50)





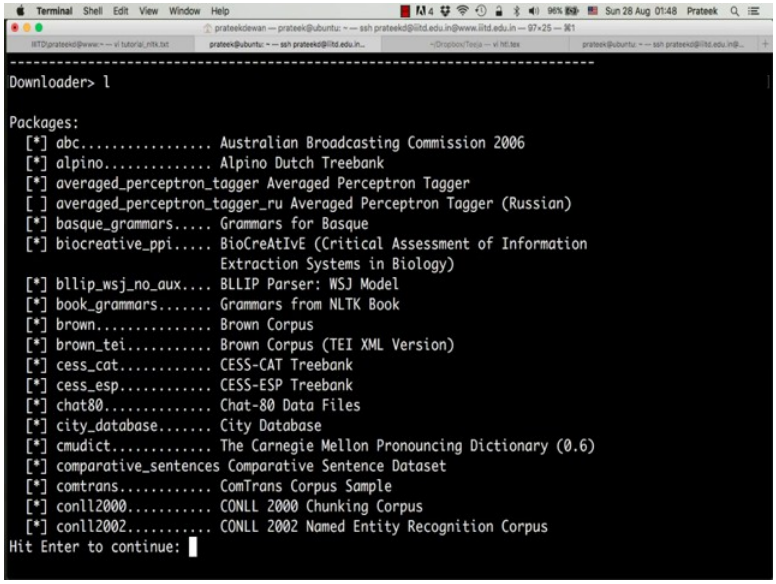
Now, this entire corpus is about 10 GB in size. So, if you do not have enough bandwidth or have slow internet, you can just download the packages that are required. So, for example, for this particular example, if you scroll back, you can see that you needed something called; you needed a package called punkt. Now, if you scroll down in the list that nltk showed you can locate this package here.

(Refer Slide Time: 06:20)



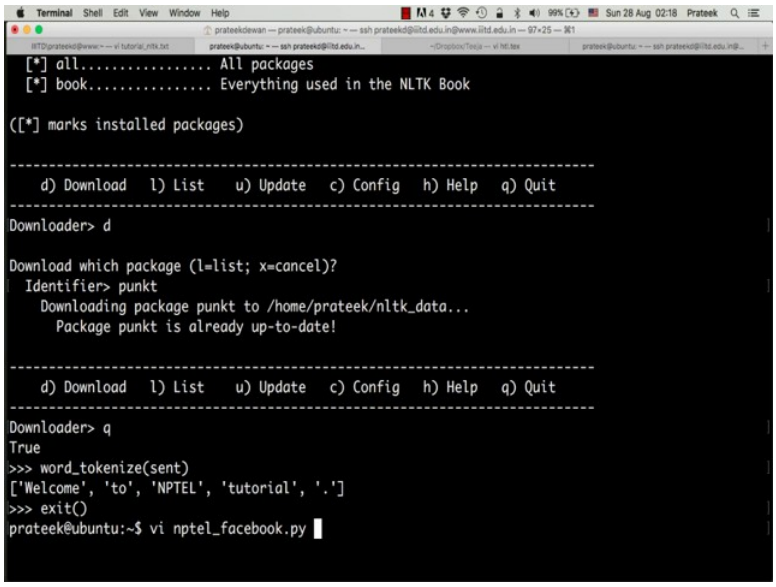
And you can type punkt instead of all to download only this model, this will just be faster, and this will save a lot of your bandwidth.

(Refer Slide Time: 06:30)



So, if you type l you can see that a star is marked in front of all the packages that we already have because we downloaded all the packages, but what you can do is just press d and type punkt to download only the tokenizer.

(Refer Slide Time: 06:36)

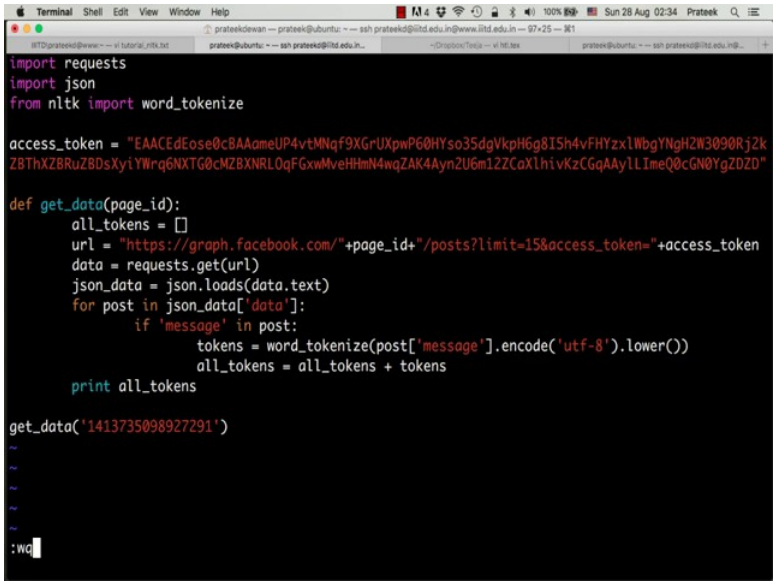


So, this says that package punkt is already up to date. Now, once you are done with downloading the corpus, just type q and press enter to exit; now if you execute that statement again, you will see that word underscore tokenize has tokenized the sentence into words and we have a list of words. Now, we will use this word tokenizer to analyze

the post that we downloaded from the NPTEL Facebook page. Type exit and exit the python shell.

Now, let us go back to our script and try to use this nltk to find the most commonly used words by the NPTEL page for their posts. So, what we will essentially do is we will go through some of the recent posts that the NPTEL page has done and we will try to find out which are the most commonly appearing words in their posts. Now, let us go back to our script.

(Refer Slide Time: 07:39)



Type from nltk import word underscore tokenize and for every message that we are iterating, what we will do is we will say tokens is equal to word underscore tokenize post message dot encode UTF-8. So, what we are essentially doing is we are converting all the posts that we have, into tokens into a list of words, now before doing that what we should do is we should also probably convert this string into lower case letters. So, we say dot lower. So, that we avoid any repetition because when we are counting word frequencies we do not want to count a word in capitals and in smalls as two different words.

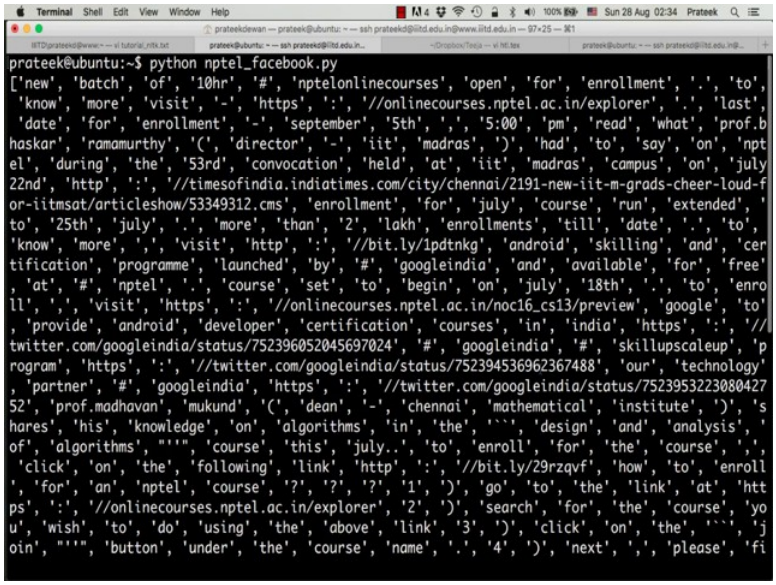
We have to maintain uniformity we will convert everything into lower case and then try to find the frequencies. Now, we have a list of words in the tokens variable for each of the posts; now we need to collect all these tokens in a single list to be able to count the frequencies. So, we will create all underscore tokens as an empty list in the beginning of

the function and every time we calculate the tokens, what we will do is we will say all underscore tokens is equal to all underscore tokens plus tokens.

So, what this will do is, this will append the tokens that we got from the current post into another list which will maintain the list of all the tokens that we have generated. So, we can get rid of these print messages and what we will now do is and let us just print all underscore tokens to see a list of all the words that are present in these posts.

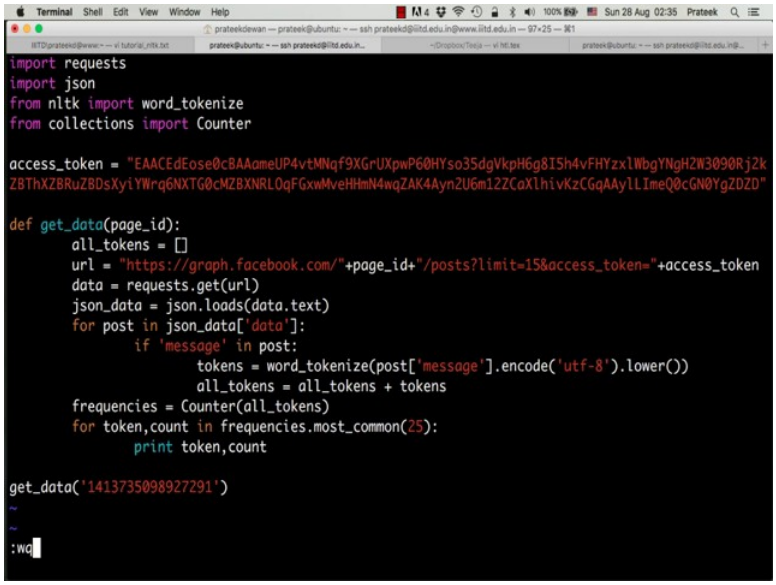
So, let us also add a limit for the number of posts that we are analyzing. So, we will say limit is equal to say 15 and access token is equal to access token. So, what we are doing is we are printing all the words that appeared in the last fifteen posts posted by the NPTEL page.

(Refer Slide Time: 10:06)



So, there you see a list of all the words that were contained in the posts that the NPTEL page did. Now, to calculate the frequency of these words, what we will do is we will use the counter function from python.

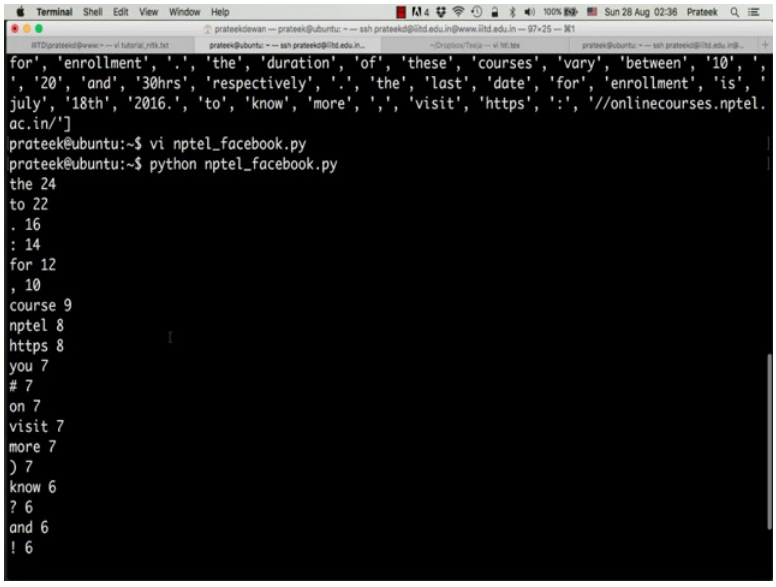
(Refer Slide Time: 10:24)



Now, say from collections import counter, what this counter does is given a list it will count the number of instances of all the unique elements present in this list. So, what we will do is, we will say frequencies is equal to counter all underscore tokens. Now, to print the token and its frequency for the most commonly occurring words, we can say, for token comma count in frequencies dot most underscore common.

So, let us look at the most common 25 words and we say print token comma count. Exit the editor and say python space NPTEL Facebook.

(Refer Slide Time: 11:22)

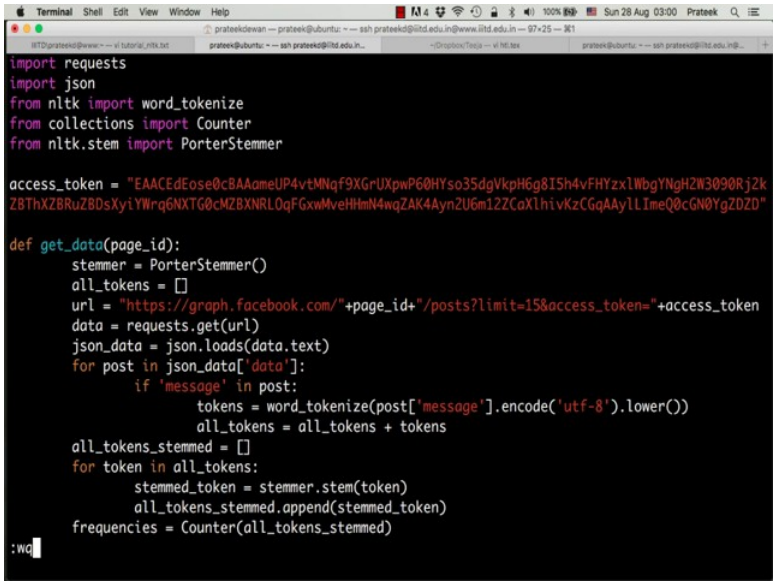


So, there you see, if you see the word "the" has been used 24 times. Similarly, "to" has occurred 22 times, there is this full stop, which has occurred 16 times; for, course, NPTEL and all of these. Now, you have an idea of what are the most common words NPTEL uses in their posts. So, as evident, the word course and NPTEL is appearing a lot of times, similarly https which means they have posted URLs containing https which are occurring a lot. If you see, there is another problem here, what we are seeing is we are seeing course and courses appear together.

Now, essentially if you notice, course and courses represent the same base word which is course. So, for example, read, reading, reader, read and all of these essentially are derived from the same word which is read; now the process of reaching this base word from any word is called stemming. We can use stemming to eliminate repetitions like this, to avoid course and courses appear separately and instead just consider them as the base word course.

Now, NLTK provides some stemming functions which you can use to convert your words into the base stem word.

(Refer Slide Time: 12:50)

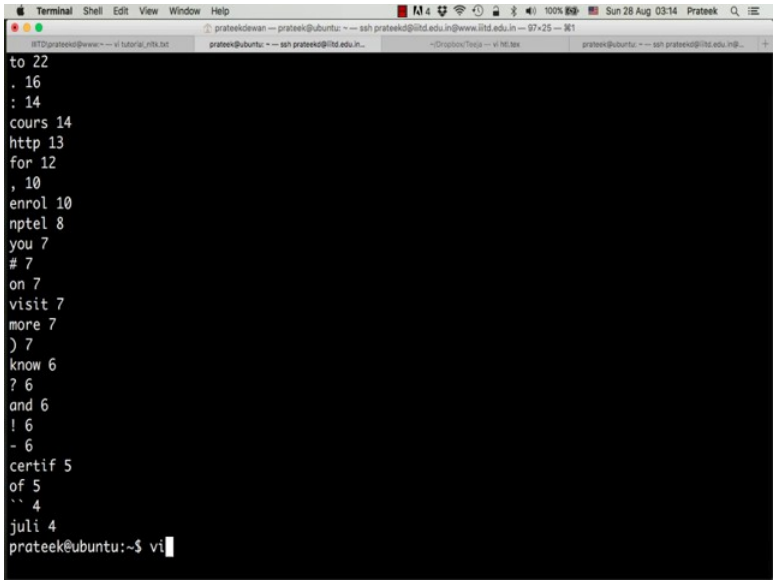


So, let us go back to our script and say from nltk dot stem import porter stemmer there are multiple types of stemmers; the one we will use is called the porter stemmer and we will create a stemmer object by saying stemmer is equal to porter stemmer and we will use this stemmer to stem all the tokens in our list. So, what we will do is let us create

another list called all tokens stemmed as a blank list and for token in all underscore tokens we say stemmed underscore token is equal to stemmer dot stem token.

By this line, what we are doing is we are converting token into it's stemmed version and storing it in a new variable called stemmed underscore token; now let us append this stemmed token into this new stemmed list that we created. So, we say all tokens underscore stemmed dot append stemmed underscore token and now for the frequencies, let us calculate the frequencies for the stemmed tokens. Now, this should solve our course and courses problem and consider them as the same base word course.

(Refer Slide Time: 14:26)

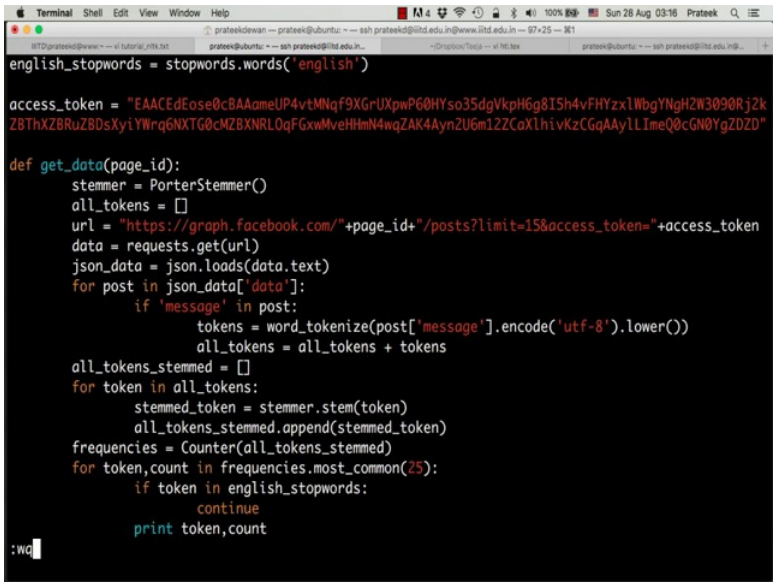


So, you see that course and courses have been combined into one word and the frequency is now fourteen; similarly enrollment has changed to enroll and certification and certify have changed to certif., this is the stem word for the words certify, certification and certificates. Now, while analyzing text, words that appear commonly in the English language are not of much help for analyzing the content that is under consideration, for example, words like and or the or to or for are words which appear very frequently in the English language naturally. So, one good way to perform better analysis is to find a way to eliminate these words; now in all languages the most commonly appearing words in the language are known as stop words.

Now, NLTK provides stop words for the English language which we can use to identify these words and eliminate them from our analysis. So, let us go back to our script and

say from nltk dot corpus import stop words.

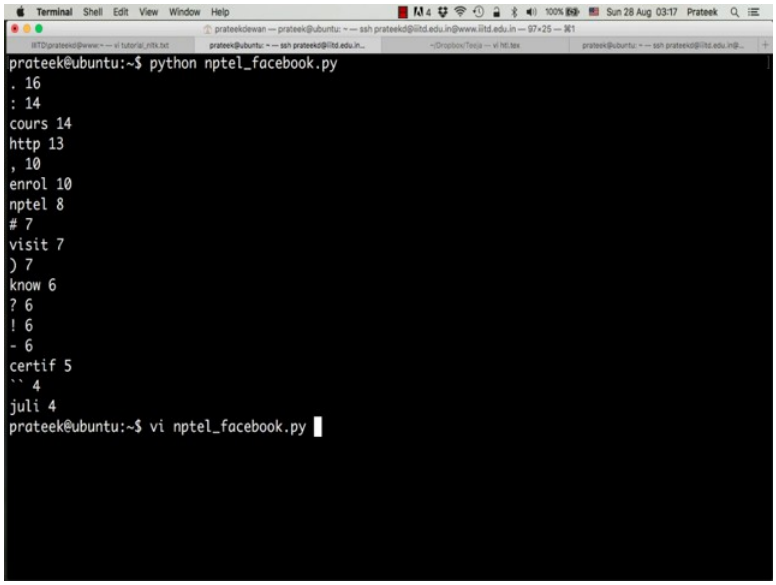
(Refer Slide Time: 15:30)



Now, let us say english underscore stopwords is equal to stopwords dot words english. Now we have a variable called english underscore stopwords which contains english stopwords that we have obtained from the nltk corpus.

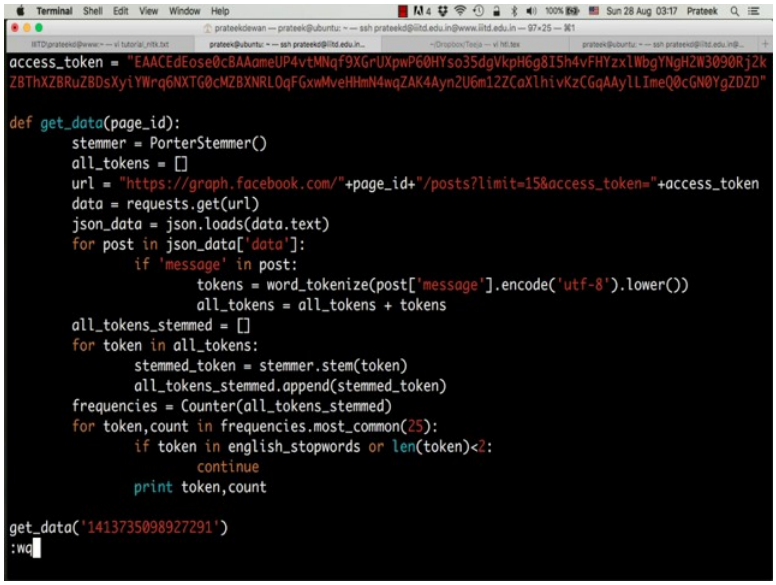
Now, while printing our output, let us put an if condition saying if token is present in english underscore stopwords, continue; that is if the token is present in the list of english words, this print statement will not execute and the continue statement instructs the for loop to go to the next iteration without executing the rest of the for loop body. So, essentially whenever the for loop encounters the stopword it will not print it.

(Refer Slide Time: 16:42)



So, let us execute this code and see what happens. So, you see the most commonly appearing words like the, and, for and to have disappeared from this list and this gives us a better sense of the content that we are analyzing. To further refine our output we can probably eliminate these punctuation marks and special symbols as well. Let us go back to the script.

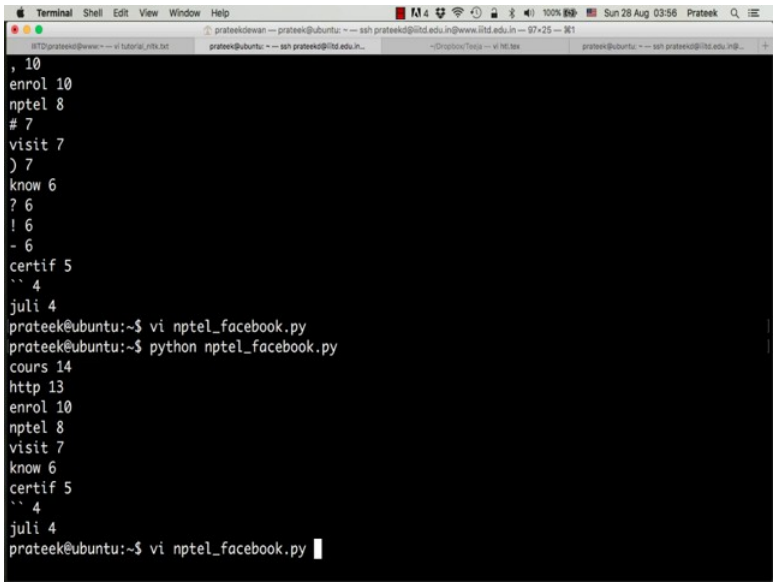
(Refer Slide Time: 17:13)



Say if token in english stopwords or length of token is less than two characters which means any token which is one character will not appear in the output. So, this will

eliminate full stops, parenthesis, hash and all those kinds of tokens.

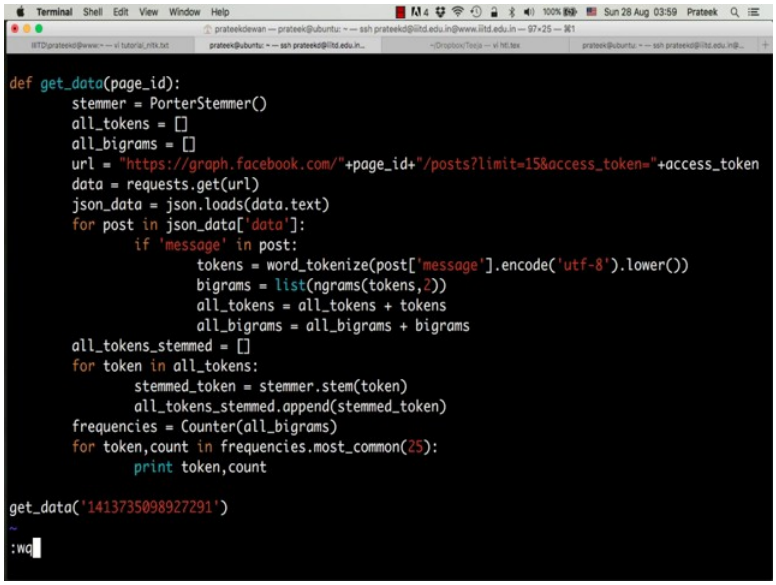
(Refer Slide Time: 17:30)



Now, let us execute this code and now you see the difference in the output. So, this gives us a much better refined analysis of the content that the NPTEL page is posting. We can also look at bigrams and trigrams and other n grams to analyze this content even better. So, right now we have looked at only singular words that are appearing most commonly. Now in some cases it might be more helpful to know what are the pair of words which are occurring together or what are say 3 or 4 or x number of words which are appearing together most frequently.

For example, names. If you consider a full name for example, Barack Obama and you look at the most commonly appearing words in text which contains Barack Obama, we will notice that Barack and Obama will appear separately, but what if you want to know the number of times Barack Obama appeared together or for that matter, any pair of words that have appeared together. For that we will use an NLTK package called n grams.

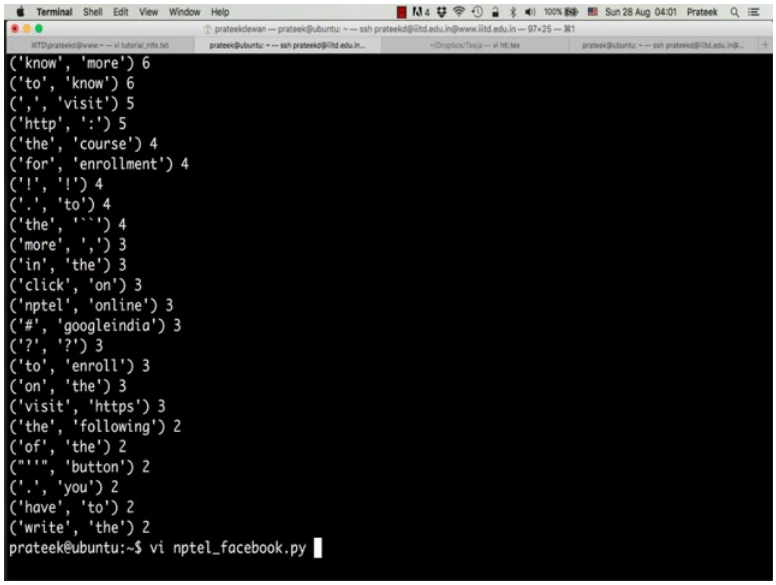
(Refer Slide Time: 18:36)



So, let us go back to the code and say from nltk import ngrams. Now, let us create another list called all underscore bigrams where we will append all the bigrams that are appearing. Once, we have the tokens, all we need to do to get bigrams is to say bigrams is equal to n grams tokens comma two this line will calculate bigrams and store them in a list called bigrams; to obtain trigrams we can simply change this number two to three and the ngrams function will then return trigrams instead of bigrams. We also need to convert the output of this ngrams function into a list. So, we say list ngrams tokens comma 2.

Now, as we did for tokens, we also need to append these bigrams to the all bigrams list. So, we say all underscore bigrams is equal to all underscore bigrams plus bigrams and now we directly go to the counter function to count the frequency of all underscore bigrams and we can get rid of this stopwords and token length condition.

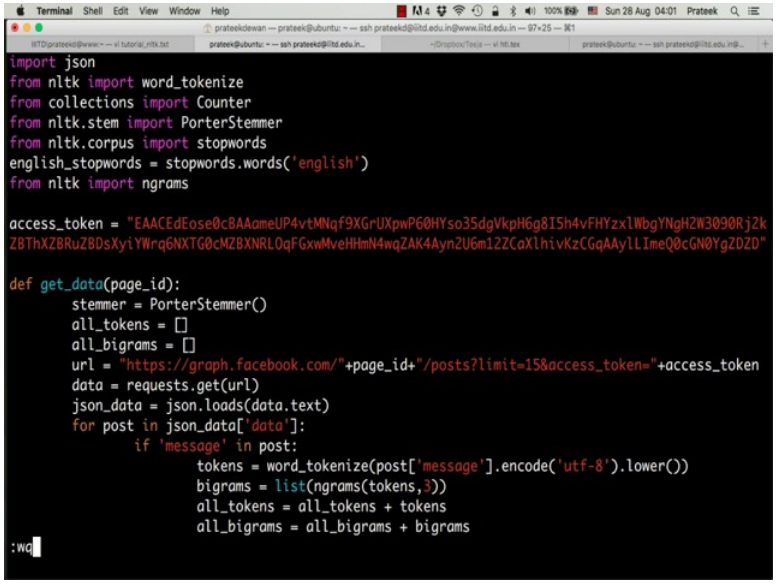
(Refer Slide Time: 20:36)



There you go, you see that https with a colon is the most frequently appearing bigram in this data set followed by know more or to know and so on. If you compare this output with the output of the uni grams, you will see considerable differences.The bigram output clearly tells us that NPTEL posts a lot of posts containing phrases like 'to know more', 'visit URL' or 'for enrollment' or 'nptel online' or 'hashtag googleindia' and so on. In the unigrams output, we were not able to identify such phrases.

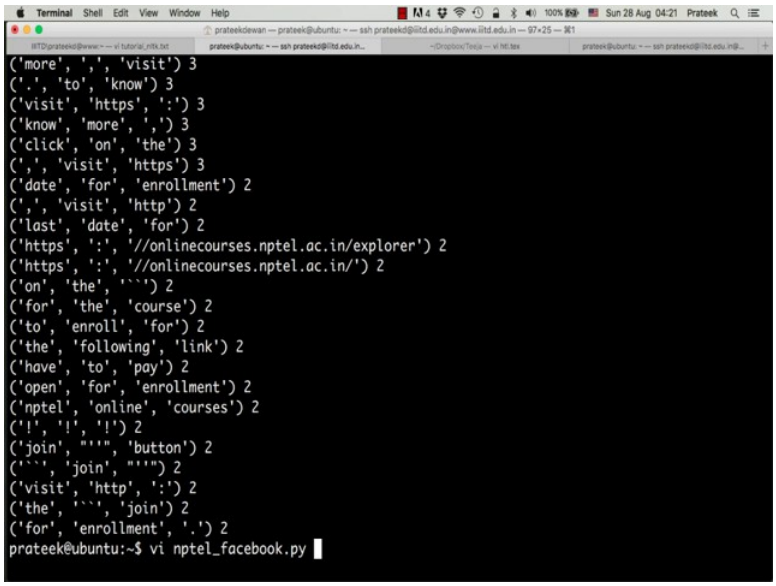
Now, let see what are the most commonly appearing trigrams. So, let us go back to the code and just replace this 2 by 3 and that is all we need to do.

(Refer Slide Time: 21:03)



Run this code again and you see the most commonly appearing trigrams.

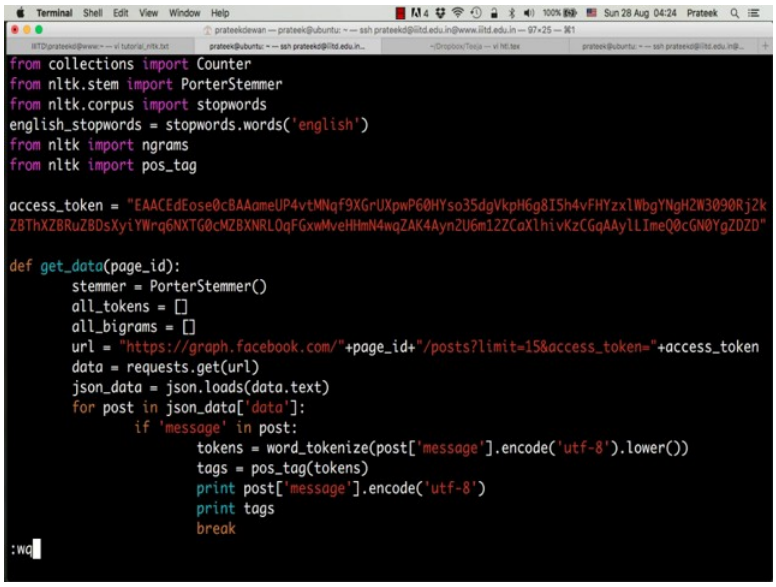
(Refer Slide Time: 21:08)



So, like we noticed - to know more comma visit and visit http click on the date for enrollment and other similar phrases which are appearing very frequently in NPTEL posts. Another interesting aspect of social media content that can be studied is the part of speech tags or the sentence structures that people use while they are posting online, for example, it might be very helpful in some cases to identify nouns, pronouns, verbs, adverbs and all such words in a sentence automatically. So, let us see how to do that

using nltk.

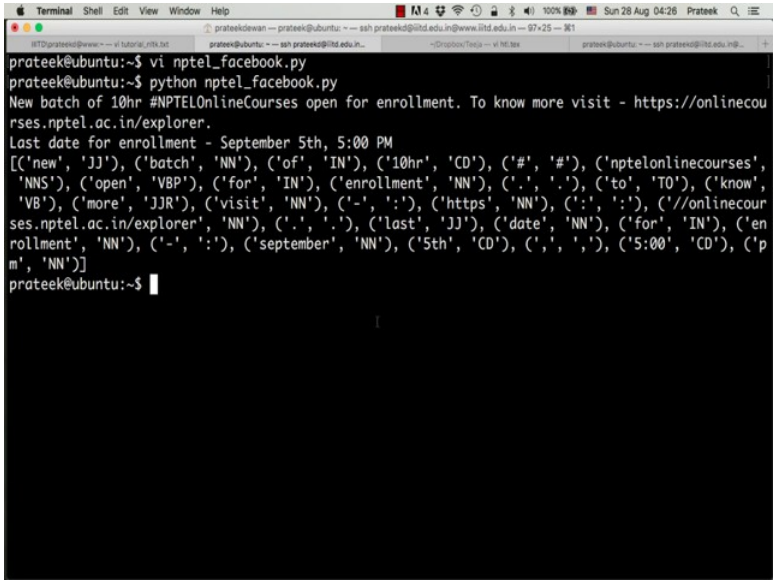
(Refer Slide Time: 21:58)



Go back to the script and say from nltk import pos underscore tag. Pos underscore tag stands for part of speech tagging. This package is capable of parsing a sentence as tokens and marking each token with a part of speech associated with it.

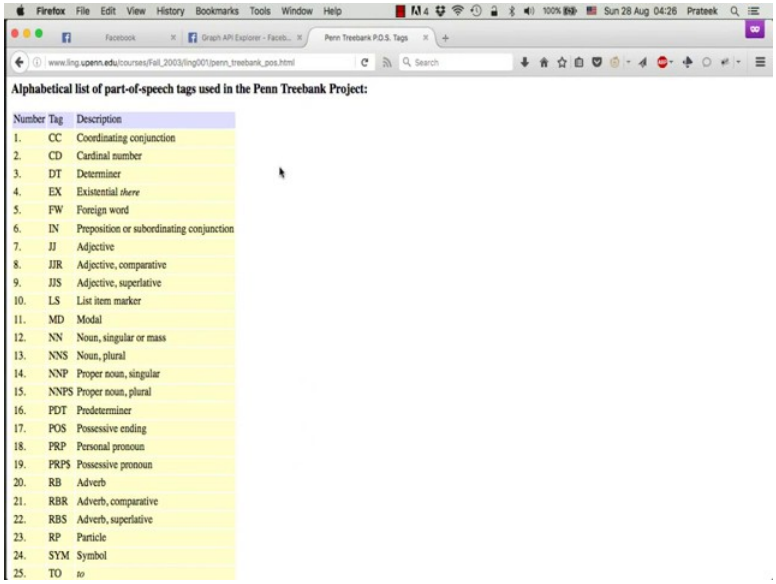
So, let us see how this works, as an example we will look at the part of speech tags for only one post to make it easier for us to understand. So, after we have obtained the tokens, we say tags is equal to pos underscore tag, tokens, print tags and remove the remaining part of the code. In addition to tags, let us also print the message to avoid confusion to see exactly what the post is and what are the tags associated with each word in the post. So, we say print post message dot encode UTF-8 and let us break this loop here, so that we can focus on only one post for the output.

(Refer Slide Time: 23:22)



So, as you see here the message reads new batch of ten our NPTEL online courses open for enrollment. Now the part of speech tagger has taken each word into consideration and assigned a part of speech tag for each of these words. The explanation for each of these tags is available online at this URL.

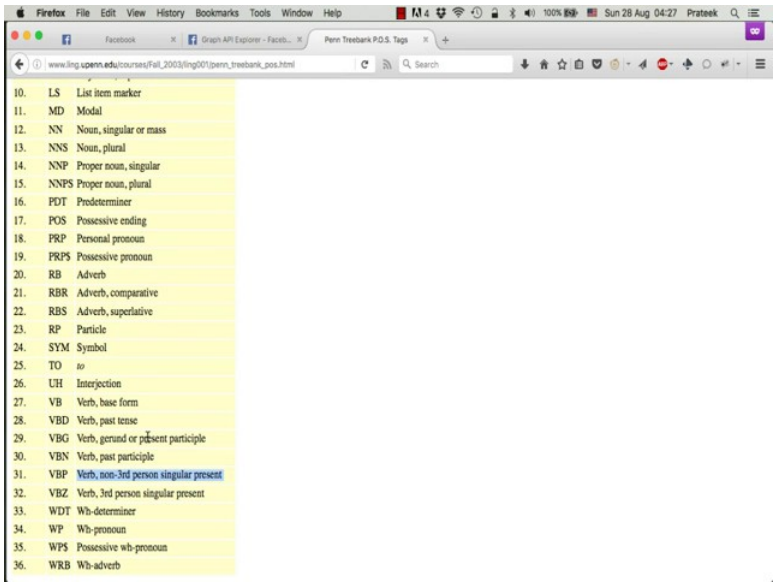
(Refer Slide Time: 23:47)



So if you notice, the first word is marked with the tag jj, going back to the lookup table, jj refers to an adjective, which means new is an adjective. Similarly, batch is a noun. of is a preposition. 10 hour is a cardinal number. open is a non third person singular present

verb and so on.

(Refer Slide Time: 24:30)



10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	to
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

Part of speech tags can be very helpful in understanding the structure of a sentence. They can also be used to identify whether a sentence follows correct grammar or not. The sentence structure and grammar can be used as a feature to differentiate between different kinds of online social media users. That is all for this tutorial.

Let us just go back and revise what all we covered in this tutorial. In terms of nltk, we looked at how to tokenize words and convert sentences into a list of words. We also saw how to calculate the frequency of these words, then we looked at stemming which converts words into their stem words. Then we looked at the nltk corpus for english stopwords and also looked at how we can calculate bigrams and trigrams using the ngrams package. Finally, we looked at part of speech tagging available as part of the natural language toolkit in python.