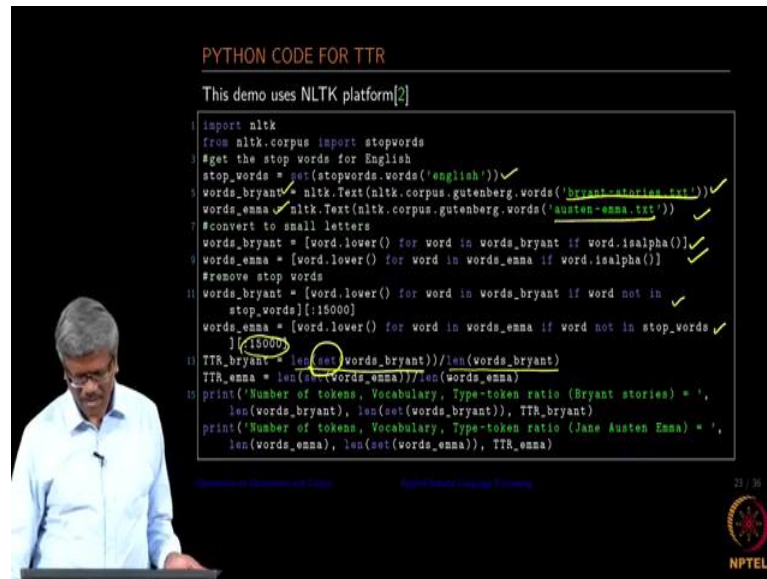


Applied Natural Language Processing
Prof. Ramaseshan Ramachandran
Department of Computer Science and Engineering
Chennai Mathematical Institute, Madras

Lecture – 10
Statistical Properties of Words Part 03

(Refer Slide Time: 00:15)



```
PYTHON CODE FOR TTR

This demo uses NLTK platform[2]

1 import nltk
2 from nltk.corpus import stopwords
3 #get the stop words for English
4 stop_words = set(stopwords.words('english'))
5 words_bryant = nltk.Text(nltk.corpus.gutenberg.words('bryant-stories.txt'))
6 words_emma = nltk.Text(nltk.corpus.gutenberg.words('austen-emma.txt'))
7 #convert to small letters
8 words_bryant = [word.lower() for word in words_bryant if word.isalpha()]
9 words_emma = [word.lower() for word in words_emma if word.isalpha()]
10 #remove stop words
11 words_bryant = [word.lower() for word in words_bryant if word not in
12 stop_words][:15000]
13 words_emma = [word.lower() for word in words_emma if word not in stop_words
14 ][:15000]
15 TTR_bryant = len(set(words_bryant))/len(words_bryant)
16 TTR_emma = len(set(words_emma))/len(words_emma)
17 print('Number of tokens, Vocabulary, Type-token ratio (Bryant stories) = ',
18 len(words_bryant), len(set(words_bryant)), TTR_bryant)
19 print('Number of tokens, Vocabulary, Type-token ratio (Jane Austen Emma) = ',
20 len(words_emma), len(set(words_emma)), TTR_emma)
```

So, I am going to be showing another demo that computes the value of TTR again I am going to be using the NLTK platform to perform this operation. in this case we are going to be taking two corpus one is Bryant stories corpus, the second one is Austen Emma corpus. We want to find out the lexical variety of both Bryant stories and the Austen Emma text. As in the previous case I am getting the words I have two different word set now one is for Bryant another one is for Emma I am also using the stop words, so remove all the stop words from the word collection.

So, as in the previous case I am converting all the words into small letters and making a collection for both Emma as well as for Bryant and then I remove the stop words for both Emma and Bryant ok. So now, what I am going to do is I am going to find the TTR ratio for Bryant and Emma. in this case the lengths of both Austen Emma text and Bryant stories text very ok. we cannot be compared to two different word lengths.

So, I am going to using the count up to 15000, for example for every 15000 words let us find out what the TTR ratio is, ideally how is it done is it is done for every 1000 words

you take 1000 words of Bryant 1000 words of Austen Emma corpus and then perform the TTR on both and then see what is the lexical variety in both this text. in this case you know for the sake of this demo I have taken 15000 words ok, I have taken the first 15000 words and then trying to find out what is the TTR value for both Bryant corpus and Emma corpus.

So, when I perform the operation you can see that the unique vocabulary is represented by the set here ok. when you have used the operation set to automatically removes all the duplicate words and then keep the only unique set as the vocabulary and then you get the total number of word count in the given corpus ok.

(Refer Slide Time: 03:01)

TTR DEMO - RESULTS

It is not reasonable to compare two unequal sized documents. A standardized TTR is used for fair comparison where the token size is restricted to the first 15000 tokens

	Number of tokens	Vocabulary	Type-token ratio
Bryant stories ✓	15000	2796	0.19
Jane Austen (Emma)	15000	3274	0.22 ✓

NPTEL

When you do this operation what you have in the next slide is a table that represents the vocabulary and the type-token ratio for each of the corpus. As I mentioned earlier it is not reasonable to do or compare two unequal sized documents that is why I have restricted the token count to 15000 in these cases.

So, if you look at the vocabulary since this is the same we have the vocabulary count of 2099 2796 and 3274 for Jane and then if you look at the type-token ratio for Bryant stories 0.19 and if you look at this it is 0.22 it is almost close, but Jane Austen has used more vocabulary than Bryant stories. this is used to compare two different English literature texts and find out who had used better vocabulary in a sense in terms of the number of vocabulary used all right ah.

(Refer Slide Time: 04:18)

APPLICATIONS OF TTR

- Monitor the vocabulary usage
- Monitor child vocabulary development
- Estimate the vocabulary variation in the text

25 / 36

NPTEL

So, what is the application of this? So as I mentioned earlier it is used to monitor the vocabulary usage, you can monitor the child's vocabulary development, you can also estimate the vocabulary variation in a given text.

(Refer Slide Time: 04:37)

INVERSE DOCUMENT FREQUENCY

In order to attenuate the effect of frequently occurring terms, it is important to scale it down and at the same time it is necessary to increase the weight of terms that occur rarely.

Inverse document frequency (IDF) is defined as

$$IDF_t = \log \left(\frac{N}{D_{ft}} \right) \quad (8)$$

where N is the total number of documents in a collection, and D_{ft} is the count of documents containing the term t

- Rare documents gets a significantly higher value
- Commonly occurring terms are attenuated
- It is a measure of informativeness
- Reduce the tf weight of a term by a factor that grows with its collection frequency.
- If a term appears in all the documents, then IDF is zero. This implies that the term is not important

26 / 36

NPTEL

Continuing on the discussion on the important terms in natural language processing, now we will take up inverse document frequency. Earlier we spoke about the term frequency where we try to count the number of occurrences of a given term in a given document. counting the terms does not really give you the sense of what really is are present in a

given document. you like to extend that little further in terms of adding the new technique to find out the relevance of a given document.

So, in this case we are going to be finding the inverse document frequency across the corpus. let us first define what inverse document frequency is and then move onto the examples and how it can be utilized to find the ranks of the documents in the corpus for a given word and so on ah. We also need to attenuate the frequency of a term that is occurring very frequently. if you have too many of such terms you know they are going to be curtailing the relevance of the other terms in the given document.

So, we want to really make sure that there is some kind of normalization across the corpus done so that the retrieval of documents for a given term is always in the normalized fashion. Let us first define what IDF is; IDF is defined as the log of the ratio of the number of documents in a collection to the document frequency. Let us first define each one of that N is the total number of documents in the collection and the D_f is the document frequency, which means it is the count of the documents containing the given term t .

So that means, IDF is for every term in the given document, by doing IDF rare terms gets a significantly higher value. For example, there are a thousand documents and then a particular term occurs only once in the entire corpus, which means the IDF value is higher, and if you have a term appearing in every document then the IDF value for that particular term is lower. in doing so we are actually attenuating the commonly occurring terms, it also now brings into the concept of informativeness in the document. What I mean that is even if there are terms that are not really appearing in all the documents we still have some relevance to those particular terms in the given corpus.

It actually reduces the TF rate of a term by the factor that grows with the collection frequency. Suppose if you have a term with larger frequency and if it appeared in all the documents across the corpus the TF IDF if you multiply both you will have a normalized small value. otherwise if you only consider the TF the value of TF would be very high ok. by extending this to the terms where that particular term appears in every document in that collection then IDF would be zero, which means the term is not very important to the retrieval operation ok.

(Refer Slide Time: 08:25)

TF-IDF

Composition of TF and IDF produces a composite scaling for each term in the document

$$t f-id f_{t,d} = t f_{t,d} \times id f_{t,d} \quad (9)$$

- The value is high when t occurs many times within a few documents
- The value is very low when a term appears in all documents

NPTEL

So, as I mentioned earlier of we have to combine the TF and IDF that is the ideal situation in many corpora we will do the TF and IDF composition to find the value of a term, so this is a very commonly used information retrieval. in this case the value of a term is very high if it occurs within a few set or with occurs within a small set of documents, the value is very low when the term appears in all documents. So that means, TF IDF is a very standardized term or the technique that we want to use in the corpus collection ok.

(Refer Slide Time: 09:19)

INVERSE DOCUMENT FREQUENCY-IDF

IDF is the inverse frequency of the word 't' appearing in the corpus. It is computed as

$$IDF \text{ of a term } t = \log_{10} \left(\frac{\text{Total number of documents in a corpus}}{\text{Count of documents with term } t} \right)$$

IDF is the measure of **informationness**

Example:
Consider a corpus with 100000 documents. The word **moon** occurs in some documents (say, 100) with the following frequency:
 $TF_{d_1} = \frac{20}{42}, TF_{d_2} = \frac{30}{250}, TF_{d_3} = \frac{20}{250}, TF_{d_4} = \frac{5}{125}$ and $TF_{d_{1000}} = \frac{20}{1000}$
The total number of words in the corpus = 100000

$\therefore IDF_{d_1} = \log_{10} \left(\frac{100000}{100} \right)$
 $TF_{d_1} * IDF = 0.141$

If the word **Andromeda** appears only once d_1 , then $TF_{d_1} * IDF = 0.0117$ If the word **the** appeared in every document and 45 times in d_1 , then $TF * IDF = 0$

NPTEL

Let us take one small example and then explain the importance of TF IDF. just explaining rather expanding the definition with the with a sentence there, IDF of a term t equal to $\log_{10} \frac{I}{N}$ I am taking the base as ten here the total number of documents in a corpus to the count of documents with the term t . IDF is now the measure of the informativeness as I mentioned earlier. if we consider a corpus with thousand terms and we assume that there is a word moon that occurs in some document, let us say in 100 documents with the following frequency.

So, in document one, it appears about 20 times and in a document 1000, it appears about 20 times and then rest of the numbers if you add up all those frequencies it may not be 100, in this case, rest of them probably would be appearing in other documents as well in small numbers. if you calculate the IDF for the term moon here it will be the logarithmic value of 100000 by 100 which means it is equal to 3 and then the term frequency and IDF would be equal to 0.141. If you do not consider the TF IDF and only consider TF the word Andromeda if it appears only once in the entire corpus it will have no relevance at all. Because if you assume that it appears only once in one document let us say d 1 then it is going to be 1 by 427.

So it is of very less significance, but when you consider the IDF for this it would be equal to 5 right. The IDF is going to be 5 and then the TF IDF is going to be 0.0117 it is here. if the word d appeared in every document then the TF IDF will be equal to 0, which means the entire word d will the will not be considered at all in the relevancy of documents ok.


(Refer Slide Time: 11:56)

DOCUMENT RANKING USING TF-IDF

Using the TF-IDF, the rank order for the documents can be determined for the documents for the term moon.

Document Name	tf	tf-idf	Rank
d1	0.047	0.14	3
d2	0.012	0.36	1
d3	0.08	0.24	2
d9	0.04	0.12	4
d1000	0.02	0.06	5

29 / 36



So, now let us find out for the given word moon, what are the ranks, or the rank order of all the documents depending on the TF IDF. this is one of the applications of TF IDF using which you can find the ranks of the documents based on the word moon here right. what we are seeing in the table here is the TF IDF for the word moon for documents 1, 2, 3, 9 and 1000. if you consider these five documents the ranking would be like this the one with the highest-ranking will be termed as rank 1. if you look at that you know the ranking would be ordered with respect to the TF IDF in the descending order.

So, when you display the results you start displaying this as the 1st document, this as the 2nd document, 3rd, 4th and 5th and so on ok. using TF IDF it should be possible for you to rank document for a given word and this rank is going to change if you change the word to something else.

(Refer Slide Time: 13:22)

ZIPF'S LAW

Zipf's law states that for a given some corpus, the frequency of any word is inversely proportional to its rank in the term frequency table[3]

$$f(r) \propto \frac{1}{r^\alpha} \Rightarrow f(r) \cdot r^\alpha = K \quad (10)$$

where $\alpha \approx 1$, r is the frequency rank of a word and $f(r)$ is the frequency in the corpus. The most frequent word will have the value 1, the word ranked second in the frequency will have $\frac{1}{2^\alpha}$ the word ranked third in the frequency will have $\frac{1}{3^\alpha}$ etc

Distribution of terms/words

This empirical law models the frequency distribution of words in languages. This distribution is observed across several languages with a large corpus. It may not be good enough to fit the frequency linearly, but enough to approximately model word frequencies

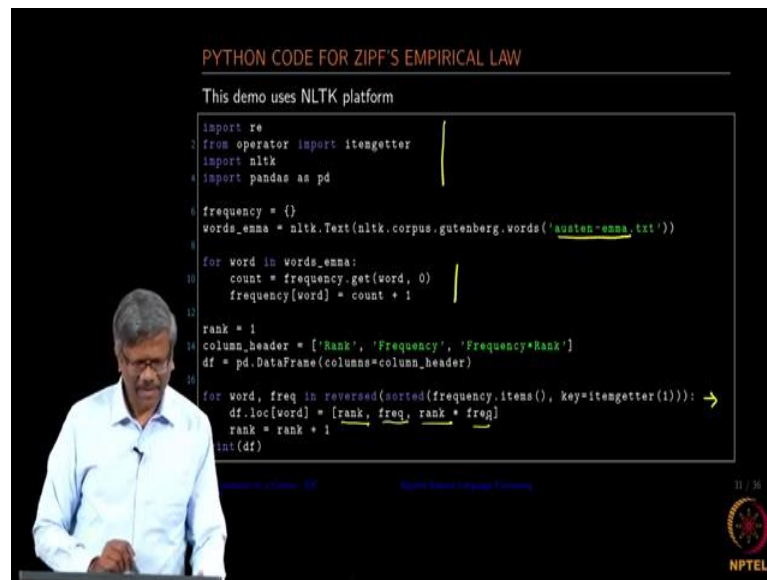
NPTEL

So, let us look at another empirical law in natural language processing which is called Zipf's law. Zipf's law states that for a given corpus the frequency of any word is inversely proportional to its rank in the term frequency table. That means if you have a word with a very high frequency and the rank would be equal to 1 right and then if you take the second one in that rank the frequency and the rank if you multiply should be same as what you get for the first frequency and its rank.

So that means if you have this could mean that the frequency of a given term and its rank if we multiply it should give a constant. This should be the same for all the terms in the given document. Here we consider alpha to be equal to 1 and r is the frequency rank of the word and $f(r)$ is the frequency of the term in the corpus ok. The most frequent word will have the value one that is its rank and then if you look at the second one it will be represented as the frequency of that particular word would be equal to $1/2^\alpha$ and then third would be equal to $1/3^\alpha$ to the power alpha and so on. If you multiply the frequency and the rank you are going to be getting a constant value this is what Zipf's law is really stating.

Is it really true in the case this is just an empirical law it may not work out very well for every document that we find or the corpus that we find? It roughly gives you that the frequency and the rank would be almost equal to a constant value ok. This is useful in terms of modeling the corpus ok.

(Refer Slide Time: 15:45)



PYTHON CODE FOR ZIPP'S EMPIRICAL LAW

This demo uses NLTK platform

```
import re
from operator import itemgetter
import nltk
import pandas as pd

frequency = {}
words_emma = nltk.Text(nltk.corpus.gutenberg.words('austen-emma.txt'))

for word in words_emma:
    count = frequency.get(word, 0)
    frequency[word] = count + 1

rank = 1
column_header = ['Rank', 'Frequency', 'Frequency*Rank']
df = pd.DataFrame(columns=column_header)

for word, freq in reversed(sorted(frequency.items(), key=itemgetter(1))):
    df.loc[word] = [rank, freq, rank * freq]
    rank = rank + 1
print(df)
```

11 / 36

NPTEL


Let us look at a small demo that really computes the Zipf's value with respect to frequency and the rank and then see how it shows the table as. I am not going to be talking about all these probably you should be able to find some details about that on the internet. I am going to be taking only the very important that we have here ok. After collecting all the terms from the corpus we are going to be finding the frequency for every word. The frequency for every word is counted using this and then the ranks are found using the second one right and then for every word that we have found we are going to be listing by the rank it is the frequency and the rank and frequency and then finally we print that particular table.

(Refer Slide Time: 17:01)

ZIPF'S LAW - DEMO

Word	Frequency	Rank	Frequency*Rank
to	5183	3	15549
the	4844	4	19376
and	4672	5	23360
of	4279	6	25674
I	3178	7	22246
as	1387	21	29127
-	1382	22	30404
he	1365	23	31395
for	1321	24	31704
have	1301	25	32525
is	1220	26	31720
with	1187	27	32049
Mr	1153	28	32284
very	1151	29	33379
but	1148	30	34440

12 / 36



So, if you do this operation you are getting some table where the words are listed as the first column, the frequency is listed in the second and the rank in the third and then frequency and the rank in the fourth one. If you look at the frequencies with a high rank right you do not see any correlation to what Zipf's law is stating and then if you go to the ranks at the lower order you see some correlation there ok. That means commonly occurring terms in the mid-frequency cycle you have some kind of correlation that. again I am stating that this is only used for modeling the corpus and not exactly the representation of any of the corpus ok.

(Refer Slide Time: 17:55)

MANDELBROT APPROXIMATION


Mandelbrot derived a more generalized law to closely fit the frequency distribution in language by adding an offset to the rank

$$f(r) \propto \frac{1}{(r + \beta)^\alpha} \quad (11)$$

where $\alpha \approx 1$ and $\beta \approx 2.7$

It is still a wonder how such intricate language generation fits into a simple mathematical relationship. It seems so unreal and perhaps unreasonable ☹

13 / 36



So, moving onto the next empirical definition Mandelbrot approximation again is an extension of Zipf's law, where Mandelbrot studied a lot of corpora and found that there could be some approximation. that the frequency and the rank multiplication should yield to some standardized result rather than some kind of frequent measurement that we have obtained in the earlier cases.

So, he tried to modify the same Zipf's law by adding a β ok. if you look at that you know there is only one addition to this which is beta. if I make β to be equal to 0 then it is the same as the Zipf's law. if you look at these approximations or the empirical formula you will wonder that how the entire corpus could be fitted into some kind of an empirical relationship of this type, even though there are not very accurate in nature.

So, far you know we have seen the kind of operation that we can perform on a document perform on the corpus and then how we can combine the document frequency within with the inverse document frequency to find the ranks of document for a given word and so on so forthright.

(Refer Slide Time: 19:26)

HEAPS' LAW

This is used to estimate the number of unique terms M in a corpus given the total number of tokens

$$M \propto T^b$$
$$= kT^b \quad (12)$$

where $30 \leq k \leq 100$ and $b \approx 0.49$

According to this empirical law, the dictionary or the vocabulary size increases linearly with the total number of tokens/words in the corpus. It emphasizes the importance of the compression of the dictionary.

Stemming and Lemmatization

good, better, best \Rightarrow good
computer, computers, computers', computer's \Rightarrow computer

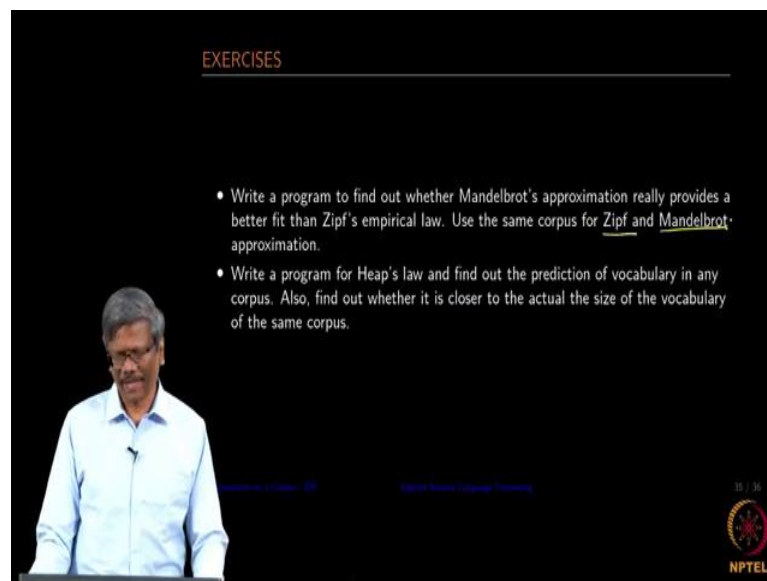
NPTEL

So, there is one more which you may want to look at which is the Heaps law. I am not going to be spending a lot of time on this, but this is again an interesting law which states that the number of unique terms is proportional to the total number of terms found in the given document.

That means according to this law the dictionary or the vocabulary size increases linearly with the total number of tokens in a given corpus. this particular law really emphasizes the importance of the compression of the dictionary that means if you have a vocabulary captured with respect to run ran running and so on. It is going to really increase the number of terms and every word be considered as a term that is not really right correct. if you want to really compress the size of the vocabulary only with respect to the root of the word and not with any other additions to that.

So, this particular law really emphasizes that by reducing those words or doing some operations on stemming and lemmatization you should have to reduce the size of the vocabulary and thereby the representation in the memory would be very minimal and so on ok.

(Refer Slide Time: 20:57)



The slide is titled "EXERCISES" in orange text at the top left. Below the title, there are two bullet points in white text:

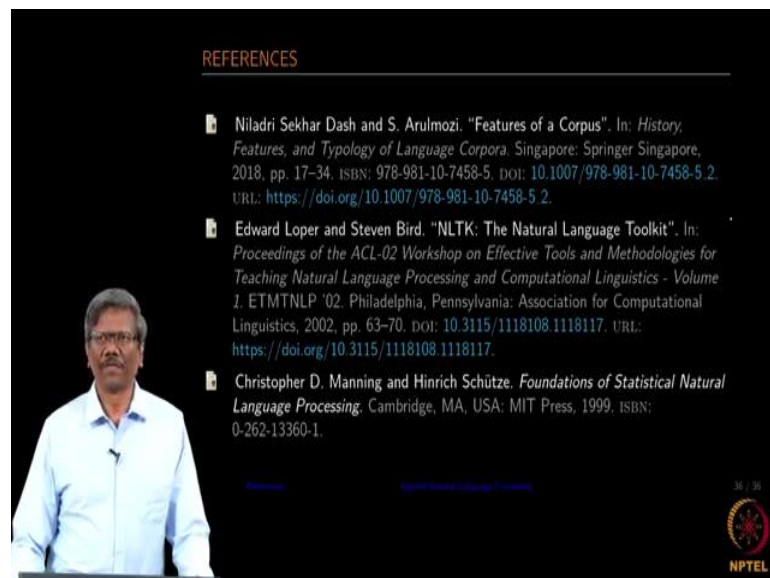
- Write a program to find out whether Mandelbrot's approximation really provides a better fit than Zipf's empirical law. Use the same corpus for Zipf and Mandelbrot approximation.
- Write a program for Heap's law and find out the prediction of vocabulary in any corpus. Also, find out whether it is closer to the actual the size of the vocabulary of the same corpus.

In the bottom left corner, there is a small inset image of a man with glasses and a light blue shirt, likely the speaker. In the bottom right corner, there is a small red and white logo with the text "NPTEL" below it.

So, with that you know the there I conclude the session with respect to what you can do with the documents, you know in terms of converting them into statistical numbers and so on, so forth. I am going to be giving you two exercises that you want to try and then make sure that you know you understand the definitions and what is really happening in all these empirical laws and so on. this is going to be about one is for the Zipf's, and another one is for the Mandelbrot. what you will have to do is you have to write a program to find out whether Mandelbrot approximation really provides a better fit than Zipf's empirical law.

So, for that you may want to use the same corpus for both Zipf's and Mandelbrot, then the second one would be to write a program for heaps law and find out the prediction of vocabulary in any corpus. Also find out whether it is very closer to the actual size of the vocabulary of the corpus ok. how do you do that this actually the second one consists of two parts one you have to write a program for the heap's law and then estimate the vocabulary that you get and second one is you have to actually use either NLTK or any other platform and then find all the words and then remove the duplicates and then only get the words in some list and then make a count of that and find out whether that count is same as what heaps law is predicting ok.

(Refer Slide Time: 22:46)



REFERENCES

- 1. Niladri Sekhar Dash and S. Arulmozi. "Features of a Corpus". In: *History, Features, and Typology of Language Corpora*. Singapore: Springer Singapore, 2018, pp. 17–34. ISBN: 978-981-10-7458-5. DOI: 10.1007/978-981-10-7458-5_2. URL: https://doi.org/10.1007/978-981-10-7458-5_2.
- 2. Edward Loper and Steven Bird. "NLTK: The Natural Language Toolkit". In: *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*. ETMTNLP '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 63–70. DOI: 10.3115/1118108.1118117. URL: <https://doi.org/10.3115/1118108.1118117>.
- 3. Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. Cambridge, MA, USA: MIT Press, 1999. ISBN: 0-262-13360-1.

38 / 38
NPTEL

Ah This slide shows the references that I have used for this lecture.