

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

Lecture – 09
Statistical Properties of Words Part 02

(Refer Slide Time: 00:15)

INCIDENCE MATRIX

Let G be a graph with n vertices (v_1, v_2, \dots, v_n) and m edges (e_1, e_2, \dots, e_m) . Then incidence matrix of size $n \times m$ is defined as

$$x_{ij} = \begin{cases} 1 & \text{if there is an edge connecting } i \text{ and } j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

It is also called vertex-edge incidence matrix and is denoted by $X(G)$

11 / 38
NPTEL

Hello again. In the last lecture we spoke about the goal of the Natural Language Processing, we saw what type of corpus that we need ideally in order to perform some natural language processing. And, then what kind of application that we could develop when we have a corpus for the training purpose and so on so forth ok. So, in this lecture we are going to continue to work on the same, but we are going to dive deeper into what kind of terminologies, or what kind of technique that we are going to be using in order to perform some operations on the corpus.

So, the first one is going to be the incidence matrix. So, I am going to give a little bit of theory first before we go into the details of what incidence matrix with respect to our corpus means ok. So, I am going to read out what is there on the slide right now. This is the definition of the incidence matrix. Let G be a graph with n vertices and m edges. Then the incidence matrix of size n by m is defined by the relationship x it is equal to 1 if there is an edge connecting i and j it is 0 otherwise. So, this is also called a vertex edge

incidence matrix. And, it is denoted by capital X capital G ok. So, this is going to be fundamental to certain operations in the corpus.

So, we need to know a little bit about this, before we dive deeper into the corpus representation of this.

(Refer Slide Time: 02:01)

BINARY INCIDENCE MATRIX

The incidence matrix corresponding to the left is given below

	e_1	e_2	e_3	e_4
1	1	1	1	0
2	1	0	0	0
3	0	1	0	1
4	0	0	1	1

$$x_{ij} = \begin{cases} 1 & \text{if the edge } i \text{ connects the vertex } j \\ 0 & \text{otherwise} \end{cases}$$

17 / 38
NPTEL

A graphical representation of the same definition is given here as an example. So, we have 4 edges and 4 vertices 1 2 3 4 and there are edges e_1 e_2 e_3 e_4 here right. So, what we have marked here is for e_1 edge it connects 1 and 2. So, you will see e_1 having a value of 1 2. So, if you go to the 3 and 4 edges is not related to that. So, we have a 0 here right.

So, in the same fashion if you look at e_2 , it connects 1 and 3. So, you have a 1 marked for this vertex and this. So, in the same fashion if you look at e_3 and e_4 ; e_3 is connecting the vertex 1 and 4 right. And, e_4 is connecting the vertex 3 and 4 ok. So, now, we know what is the incidence matrix.

(Refer Slide Time: 03:23)

TERM-DOCUMENT BINARY INCIDENCE MATRIX

To build a term-document binary incidence matrix, let us consider Shakespeare's plays as our corpus. The terms are the vertices and the names of the plays are considered as edges. This incidence matrix does not represent any information related word order or its frequency[3]

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello
antony	1	1	0	0	0
brutus	1	1	0	1	0
caesar	1	1	0	1	1
calpurnia	0	1	0	0	0
cleopatra	1	0	0	0	0
...
...

$$x_{td} = \begin{cases} 1, & \text{if the word } t \in d \\ 0, & \text{if } t \notin d \end{cases} \quad (2)$$

13 / 36

NPTEL

And, let us see how we can translate this into our corpus. So, now, what I am doing here is I have taken the plays, Shakespeare plays in particular as my corpus and I have listed the play names here ok. And, I have used the tokenization algorithm and found all the words in all the places that I have listed here. So, they are listed along with this ok.

So, every word is listed in this. So, let us take an only certain name that we know very well and then see how we can fill this up. So, when I take the word Antony, then I want to find out whether this name appears in all the plays. So, here the one that we have mark means the name has appeared in Antony and Cleopatra.

So, the name again appeared in Julius Caesar, it does not appear in the Tempest, Hamlet, and Othello, in the same fashion for every word we fill in the matrix as 1s and 0s. So, what do you see in this you see lots of 1s and lots of 0s ok? So, in a corpus of the size of an internet size and considering you have about a trillion document which is listed as in the plays and the vocabulary listed along this axis, and not all words appear in every document correct.

So; that means, the words that appear in the document would be represented by 1 and in those document where it does not appear, they are all represented by 0; that means if you have a 1 billion word vocabulary and 1 trillion documents most of the cells would be 0; that means, we have a sparse matrix there. So, that is the situation when you consider a huge corpus.

So, in this case we have taken a very small corpus. So, we will find still many 0s in this matrix as well. So, again if you look at this term document binary incidence matrix, what this is you called as a term and this is your document ok.

So, we have filled in the document with 1s and 0s whenever the word or term is found in that particular document again we do this for the second one, the third one, fourth one and how many of our document that you have you do the same operation. So, again I am taking the idea of what we described in the theory, we have created a matrix, which is called a term document binary incidence matrix ok. So, this is how you create. So, if you want to find out how this is done you want to know more about this you may want to go to the reference that I have given in the slide, that reference number 3 ok.

So, the reference is available at the end of this presentation. So, we know this right I have I also want to explain this word term. The term here means a word or a combination of words or sometimes could be 3 you know it a phrase referencing a term a word can also represent a term. So, I would use this word interchangeably in terms of word or term and throughout the lecture ok. Let us move on what can I do with this? So, I have created a matrix. So, I need to perform some operations on this matrix. So, what kind of operation I can do on a binary matrix?

So, you know now that these are all binary representations. For example, Antony is represented as a binary vector along this direction right. A Brutus is represented as a binary vector along this direction, Cleopatra is represented as a binary vector along this direction. So, now, we have binary values. So, now, it is possible for you to perform some binary operations on this ok.

(Refer Slide Time: 07:53)

IR USING BINARY INCIDENCE MATRIX

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello
antony	1	1	0	0	0
brutus	1	1	0	1	0
caesar	1	1	0	1	1
calpurnia	0	0	0	0	0
cleopatra	1	0	0	0	0
...
...

To answer the query Brutus Caesar AND NOT Calpurnia, we take the vectors for Brutus, Caesar and Calpurnia, complement the last, and then do a bitwise AND:
 $11010 \text{ AND } 11011 \text{ AND } 10111 = 10010$. The answer for this query is found in the plays Antony and Cleopatra and Hamlet

NPTEL

So, in the previous slide we had shown how a binary incidence matrix can be constructed for a term document corpus right. So now, we want to find out if we can perform certain queries on these on this particular matrix. For example, I want to form a query about Brutus and Caesar and not Calpurnia. So, I want to find the documents where Brutus is found, and Caesar is found and not Calpurnia. So, one way to do is go look at each one of those and then find out where Brutus is found, and Caesar is found, and not Calpurnia right there is one here.

So, if I have a very small set of documents, I can do this in a manual fashion assuming I have about a trillion documents, I want to perform this operation it is similarly not possible to do that. One way to do this operation is first to take the Vectors, Brutus, Caesar, and Calpurnia, and then perform the operation. So, in this case we have Brutus, Caesar, and Calpurnia and here we have asked for not Calpurnia.

So, what you do is you do the not of this operation of this vector ok. You take the vector axis for Antony, as is for Brutus, and then complement Calpurnia, and then perform the operation. Now, you can do a bitwise and operation what you get is 10010. So, what does this represent? So, one represents here the document, the 0 represents this the last one represents Othello. So, by doing this operation, now I am able to perform a simple query where Brutus and Caesar and not Calpurnia is for ok.

So, in this way I can perform binary queries along with the document and retrieve documents where these terms are found or not found ok. This is one of the examples of the information retrieval problem. So, by applying this to various other documents using binary representation, it is possible for you to do the information recovery ok. So, let us move on to the next one.

(Refer Slide Time: 10:45)

WORDS AND TERMS

The basic alphabet for the purpose of NLP is a *word*
 The next logical step after the binary representation of words or terms t , is to assign weights to words

- The atomic unit for constructing a word in a language is its alphabet
- We use term (co-located/co-occurring words) and *word* as atomic.
- It is necessary to consider the numerical representation of the word for computation purposes
- Vocabulary of size $N = 1 \dots n$ is defined as $V = \{w_1, w_2, w_3, \dots, w_n\}$ is the vocabulary containing unique words of a language
- Some words found in V appear in documents ($D = D_1, D_2, D_3, \dots, D_m$), once or several times or may not appear at all.

m 1Bk IT

15 / 38
 NPTEL

So, now what we have seen is after tokenization words come into play. So, the basic purpose of NLP is to identify the meaning of the word that is formed in the corpus. So, what is the next step? The next logical step would be to find out how many times a particular word has occurred in a document or in a corpus. If you look at the previous case, there is no representation of the number of occurrences of a given word Antony.

So, it is all it is saying is whether that particular word is present or absent in the given document that is all it says it does not say how many times it occurred and so on. So, the next operation logical operation would be to find out, how many times that particular word had occurred in a given document or in the entire corpus. So, for us to know that we need to understand that, the word is the atomic in nature for the purpose of natural language processing ok.

So, this is the atomic unit, the word is your atomic unit and that is your alphabet in the natural language processing terminology. So, we use the term as I mentioned earlier the collocated that they occurred together or co-occurring words, they also can be considered

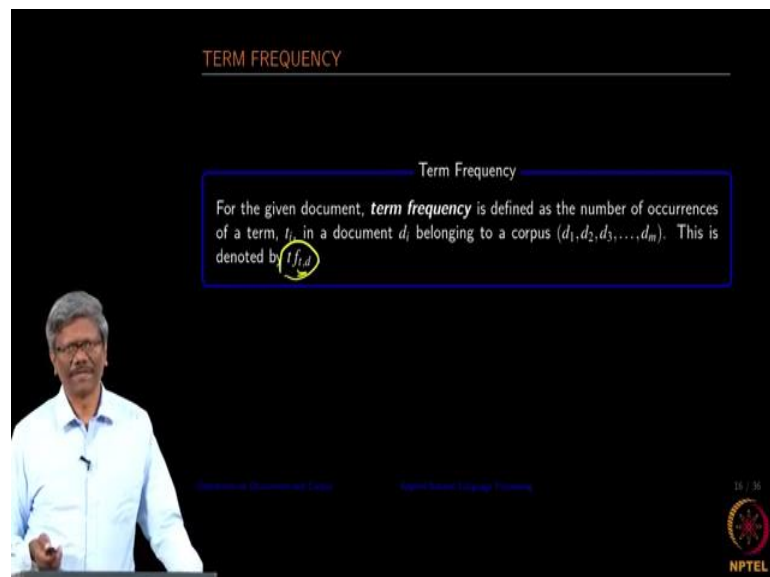
as atomic. For example, I do not want to separate New Delhi as 2 different words I want to call New Delhi as 1 term.

So, that is atomic for this purpose of natural language processing. This is why I am going to be using the word term in most situations. So, it is not just enough if you represent the presence or absence of the word, we also want to represent it in terms of some numerical quantity ok. So, we want to find out what types of numerical representation a word can have ok.

So, we will move from here to find the word count, word count with respect to the document, word count with respect to the corpus and so on. So, before that I will small definition here, we consider a vocabulary of size N for the corpus, V is defined as a unique set of words that you find in the entire corpus ok.

So, this is what you find in a dictionary. So, V is nothing, but my dictionary of words where no word is repeated ok. Some words are found in V appear in documents collection of D_1 to D_m ; that means, I have a set of documents which I represented as D_1 to D_m . And, they could vary from 1 billion to 1 trillion OKs.

(Refer Slide Time: 14:21)

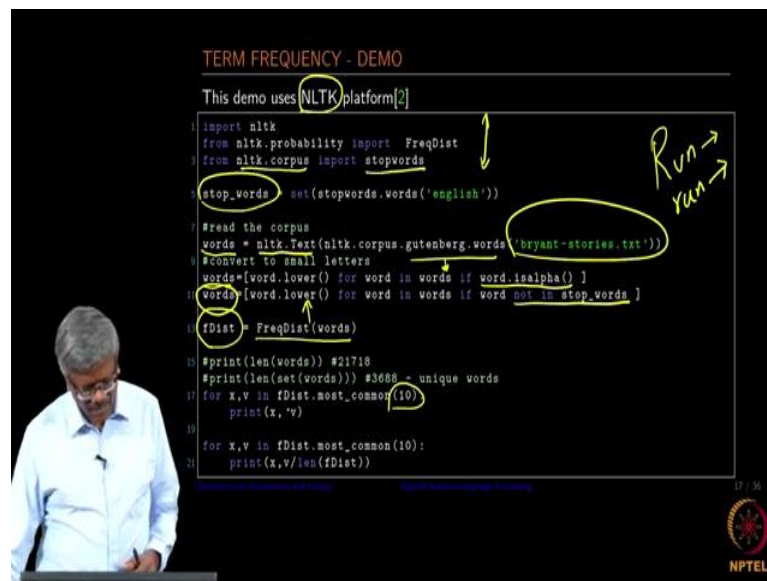


Let us now define our first interesting term in the statistical processing of the corpus. I will read out from this slide again the term frequency is defined as the number of

occurrences of a term t in a document d_i belonging to a corpus d_1 to d_m . So, this is denoted by $t f d$ ok.

So, note this suffix of these 2 OKs. So, the idea is to find out how many times a particular word occurred in a given document. Suppose, if the word occurred 100 times it will be represented $t f d$ is equal to hundred in the plain fashion ok.

(Refer Slide Time: 15:13)



```
TERM FREQUENCY - DEMO
This demo uses NLTK platform [2]
1 import nltk
2 from nltk.probability import FreqDist
3 from nltk.corpus import stopwords
4 stop_words = set(stopwords.words('english'))
5 #read the corpus
6 words = nltk.Text(nltk.corpus.gutenberg.words('bryant-stories.txt'))
7 #convert to small letters
8 words = [word.lower() for word in words if word.isalpha()]
9 words = [word.lower() for word in words if word not in stop_words]
10 fDist = FreqDist(words)
11 #print(len(words)) #21718
12 #print(len(set(words))) #3688 - unique words
13 for x,v in fDist.most_common(10):
14     print(x, v)
15
16 for x,v in fDist.most_common(10):
17     print(x,v/len(fDist))
```

So, let us take a very small example demo, where we will be finding the number of occurrences of a given term. So, this is a python program I am using the platform called NLTK, you will be able to find the details about what NLTK is and how we can install and how we can perform certain operations on the corpus that are found in the NLTK and so on on the internet. It they are very well defined and you will be able to install it yourself on your own ok.

So, I am using an NLTK for the purpose of this demo in this slide. So, what I have done here is I have written a very small program. This program actually uses a corpus called Bryant stories dot text, which is available from the NLTK platform. I am going to be using a stop word removal because certain words are not very important to me for the purpose of processing. For example, in the article the prepositions do not make any sense they do not give any meaning when you want to perform a certain statistical operation on the corpus. So, I want to remove them first. So, my list contains only certain vocabulary ok.

And, also I want to remove all the capitalization of the word so, that the capital word of R u n and run mean the same, otherwise these two will be considered as two separate words. So, in that case I need to perform the lowercase operations on all the words in the given corpus.

So, for that, you know what I have done is I made some import classes. I am importing first the nltk library and then I am using the probability library of nltk, wherein I have the frequency distribution defined. I am using the library from the corpus where stop words are defined. It should be possible for you to define your own stop words to or add whatever is available, whatever you want to add to the existing stop words ok.

So, the first thing that I do is I am just using the corpus and get all the words in the corpus as I mentioned earlier right. So, we do not have to really write your fundamental code in terms of tokenization, the platform does a lot of job for you. So, if you understand what exactly it does it will be very useful to you. So, in this particular statement actually reads the corpus Bryant stories dot text. And, then tokenize I each of the words and then put them in the collection words ok. And, then I am actually looking at whether the word is alpha or not, there could be some number there could be alphanumeric characters and so on so forth in the given corpus, I want to only look at the English vocabulary.

So, I want to find out whether a given word is an English word, then add it to my collection otherwise I ignore them. So, this particular statement goes through the entire collection of words that we have captured in this. And, then converts or creates a small subset where only the alpha words are captured ok.

And, then after capturing only the alpha word I am going to be removing all these stop words. So, not in the stop words; that means, every word which is in this stop word will be removed and only those which are not in this stop word collection will be added to the word count here ok. And, then I perform a frequency distribution of all the words that I have captured in the previous instruction ok.

So that means, my frequency distribution is again an array that contains the word and the frequencies of each of the word ok. And finally, what I am printing is I am printing the most common words that are found. So, I can print all of them it is going to be a huge collection I am restricting that to number 10 and printing them ok.

(Refer Slide Time: 20:25)

TERM FREQUENCY


Raw count of words

little	597
said	453
came	101
one	183
could	158
king	141
went	122
would	112
great	110
day	107

Term frequency adjusted to document length

little	0.1618763557483731
said	0.12283080260303687
came	0.05178958785249458
one	0.04962039045553145
could	0.042841648590021694
king	0.038232104121475055
went	0.03308026030368764
would	0.03036876355748373
great	0.02982646420824295
day	0.02901301518438178

18 / 36



Let us see what actually is the result? So, when I ran through the corpus, what I have got for the raw count of 10 words here ok. The little the word little appeared 597 times ok. And, then the word came appeared 100 and 91 times the king appeared 100 and 41 times ok, the day appeared 100 and 7 times.

So, it is a raw count of words in the given corpus alright ok. So, in this next statement running from a 20 and 21, what I have done is I have made some changes to what I did earlier right. So, earlier we only printed the word and the count. So, now, I am doing some waiting on it. So, now, I am printing the same thing the word and then, but the frequency is now divided by the size of the vocabulary.

So, that is the only chain that I am making here. So, this is very useful in terms of normalizing the values you know by providing a raw count you know it does not really give me a normalized picture whereas, in this case I am adjusting the frequency to the document length. So, they get normalized.

So, it gives me some additional weighted terms which I can use it for processing. So, we will talk about why we are doing it when we really perform certain operations. So, please remember that. So, we have to not only use we can not only use the raw count axis we can also perform some waiting on the frequencies of the words.

(Refer Slide Time: 22:15)

MULTIPLE WEIGHTING FACTORS TF

- Boolean - 0,1 ✓ (3)
- RawCount - $t_{f,i,d}$ ✓ (4)
- Adjusted to document length - $\frac{t_{f,i,d}}{M}$ ✓ (5)
- Log weighting - $\begin{cases} f_{i,d} - 1 + \log t_{f,i,d} & \text{if } t_{f,i,d} > 0 \\ 0, & \text{otherwise} \end{cases}$ (6)

19 / 36
NPTEL

There are multiple ways we can represent we saw earlier that a word can be represented either as a Boolean, whether it is present or absent represented with respect to the frequency. We can adjust it with respect to the length of the document and we can also do the log weighting of that so, that you can convert the frequency using the given formula. So, we can represent the frequency in various ways. So, these are all the different multiple weighting factors that we can apply to the term frequency.

(Refer Slide Time: 23:05)

DISADVANTAGES OF RAW FREQUENCY

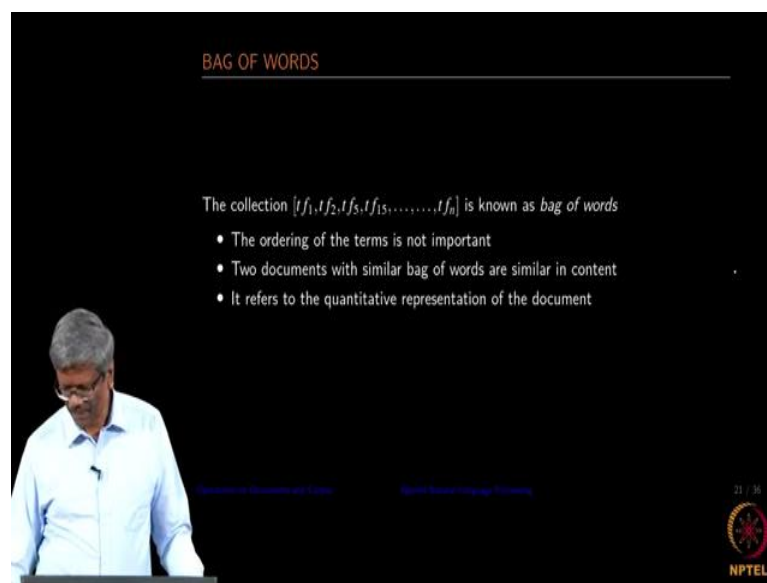
- All terms are given equal importance
- The Common term *the* has no relevance to the document, but gets high relevancy
- May not be suitable for classification when common words appear in documents

20 / 36
NPTEL

So, what are the disadvantage of or other disadvantages of having a raw frequency, you know all terms are given equal importance right. The common term you know if I had not used this stop word will have more occurrences than any other word, but it will have no relevance to the document with respect to querying and so on.

So, it may not be suitable for classification when common words appear in the document. So, if of and they appear so, any number of times they are not going to really help us in terms of classifying the document in a particular fashion ok.

(Refer Slide Time: 23:59)



The slide is titled "BAG OF WORDS" in orange text at the top. Below the title, it states "The collection $[t_{f_1}, t_{f_2}, t_{f_3}, t_{f_4}, \dots, t_{f_n}]$ is known as *bag of words*". There are three bullet points: "The ordering of the terms is not important", "Two documents with similar bag of words are similar in content", and "It refers to the quantitative representation of the document". In the bottom left corner, there is a small video inset of a man in a light blue shirt speaking. In the bottom right corner, there is a small logo for NPTEL and the text "21 / 36".

We move on to the next terminology which we will keep using in the natural language processing is a bag of words. A bag of words is nothing, but a collection of words. For example, if you have a basket and then you strip each word you know given corpus and then move it into the basket one after the other right. So, and then one, when you move what you have, is only the collection of words there is no order anything of that sort is available ok.

So, that particular operation is called creating a bag of words for a given corpus. So, this is useful when you want to perform a classification task. There is no order in the bag, you know you can any pick anything from the bag any word that appeared in the corpus might come, it does not come in any ordered sequence ok.

So, it represents a quantitative measurement for a given document. We create multiple bags and then shred those words from the given document and then move them into bag 1 bag 2 and bag 3 and bag 4 and so on. And, then if you look at the frequencies of words in each of those bags. If a certain set of words appeared in bag 1 and bag 3, then we can say that bag 1 and bag 3 are somewhat related ok. This name you can classify documents using the bag of words ok.

(Refer Slide Time: 25:47)

The slide is titled "TYPE-TOKEN RATIO" in orange text at the top. Below the title, it states: "The lexical variety of the text is defined the **Type Token Ratio (TTR)** It can be used to measure the vocabulary variation or *lexical density* of the written text and speech. The **type** is the unique vocabulary in the text which is devoid of any repetitions". To the right of this text is the formula $TTR = \frac{V}{T_n}$ where the 'V' is circled in yellow. Below the formula, it says "where V is the vocabulary and T_n is the number of tokens in the speech or written text". In the bottom right corner, there is a small red circular logo with the text "NPTEL" below it. A speaker is visible in the bottom left corner of the slide frame.

Let us move on to the next definition. This is again performed on the document so, where we want to find the lexical variety. So, this is called us type-token ratio. So, what do we do here? So, we find out what is the type with respect to the total number of words? The type is nothing, but your vocabulary the unique set of words ok, you have a total count of the number of words in a document and then you have a unique set of words right. When you create a ratio out of V and T n what you get is a TTR ok. So, this gives a very unique ratio, which tells you what is your lexical density in terms of vocabulary?

For example, if you are used a very small subset of vocabulary the value would be very small. If, you used a large collection of vocabulary, then the value would be large the TTR value would be very large. Why do we need this? It would be useful in terms of finding out how we are really improving our vocabulary on a regular basis. For example, you want to track a child's vocabulary acquisition on a regular basis, you would be able

to use this particular ratio to find out whether the child is really improving the ratio of TTR by adding more and more vocabulary to its (Refer Time: 27:33).