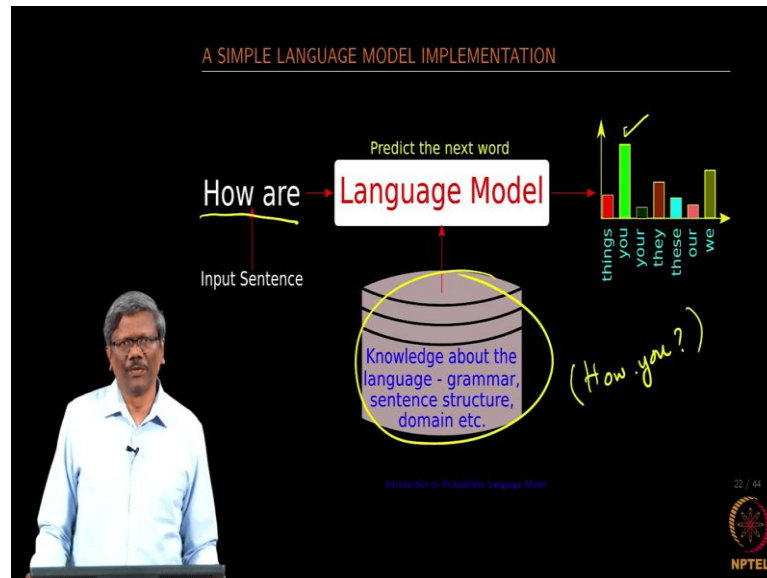


Applied Natural Language Processing
Prof. Rameshan Ramachandran
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 25
Generative Models

(Refer Slide Time: 00:15)



Hello, everyone continuing the discussion on the probability and the application of probability in the natural language processing; we are going to be talking about the language model using probability ok. We had already seen this slide earlier again for the sake of clarification I am going to be repeating this slide.

So, the language model that we are talking about using the probability contains two things; one is finding the next word or predicting the next word given the first two words or three words or four words or if you are given a sentence of trying to find out what is the probability of that sentence occurring in a given corpus or what is the probability of that sentence or what is the legality of that sentence in that particular corpus. Or suppose if there is a new word that is available as part of this sentence the probability of the sentence would become 0; if you absolutely take the product of each of the probability of the words that are found using the corpus.

So, let us going to be looking at those examples where we are going to be first finding out what is the next word and then they are going to be giving you an assignment in

terms of what is the probability of a sentence based on the corpus that you have used and then modeled the language and then later we will see how we can handle some of the words that are not in the vocabulary of the given corpus.

So, for us to do that we know that we in require a database of this type that or a model that we have built using the corpus and then given an input sentence we want to predict what could be the next word or if you are given a sentence like this ok. So, is it possible to have certain types of sentences I know I have missed are here but I am just giving as an example how you is it possible to have this sentence then what is the probability of having this?

So, these are the two different things that we would be doing based on the language model ok. So, I am going to be skipping some of the slides that we had seen earlier.

(Refer Slide Time: 02:54)

The slide is titled "TARGET AND CONTEXT WORDS". It features a speaker in a light blue shirt on the left. The main text on the slide reads: "Next word in the sentence depends on its immediate past words, known as context words". Below this is a mathematical formula:
$$P(w_{k+1} | \underbrace{w_{t-k}, w_{t-k+1}, \dots, w_t}_{\text{Context words}})$$
 To the left of the formula, there is a list of n-grams: "n-grams", "unigram - $P(w_{k+1})$ ", "bigram - $P(w_{k+1} | w_k)$ ", "trigram - $P(w_{k+1} | w_{k-1}, w_k)$ ", and "4-gram - $P(w_{k+1} | w_{k-2}, w_{k-1}, w_k)$ ". At the bottom right, there is a small logo for NPTEL and the text "28 / 44".

So, what we are going to be doing in this particular session is trying to create a model based on the corpus and then use the model to predict our next word using bigram, trigram, 4-gram or given a sentence how do we really find out the probability of that particular sentence ok.

So, in this case you know there are a few things that we have to keep in mind one is we are going to be finding the next word w_{k+1} using the context word that has happened before. So, in this case, there are few options for us one is we can just use the unigram

ok; take a whole corpus, tokenize them and then find the probability of each of the word using the method that we are described earlier and then we can use bigrams you know how to create a bigram corpus and then we can use trigram you know how to get trigrams in the corpus and again 4-gram.

So, you can go on and on ok; as we increase the number of grams the accuracy of this sentence that we are forming would be better. So, we will come we will see how that happens ok.

(Refer Slide Time: 04:20)

The slide is titled "LANGUAGE MODELING USING UNIGRAMS" in orange text at the top. Below the title, there are three bullet points in orange text: "▶ A unigram language model all words are generated independently $W_1, W_2, W_3, \dots, W_n$ and none of them depend on the other", "▶ This is not a good model for language generation", and "▶ It may generate *the the the the* as a sentence". On the left side of the slide, there is a video inset showing a man in a light blue shirt speaking. At the bottom right of the slide, there is a small red circular logo with the text "NPTEL" below it. The slide number "29 / 44" is also visible in the bottom right corner.

So, now let us look at the language model using the unigrams. So, as you know well that we are capturing the bag of words and then finding the probability of each and every word in that bag right so that means, the ordering of the word is lost. So, if you want to really generate a sentence using the unigram. So, how do we do that; so, where do we start.

So, one possible way of looking at that is to look at all the higher probability words and then start stringing them together. So, will it really work it is not the right way to really create a sentence or find the next word and so on because the sequence of the words in this sentence are lost right.

So, there is no order that is available it is just a bag of words the totally unordered set of words. So, when you look at the probability of the words and if you are going to be

stringing the word based on the higher probability if the word the would happen several times maybe this could be having the highest probability. So, if you look at the highest probably the legal sentences that could be formed using the unigram would be the as a sentence right.

But that is not the right way to form a sentence. So, it is not really possible to construct a sentence using unigram but what do we do with the unigram model. So, unigram can be used to really identify the language of the document for example, if a document contains multiple languages and you can look at each of these words and then find out whether it belongs to the language English, French, Tamil, Hindi or whatever languages ok.

So, each and every word can be used to identify the language. So, most of the time we use it for that purpose. So, unigram is not used for identifying or predicting the next word or finding the probability of an input sentence ok.

(Refer Slide Time: 06:39)

GENERATIVE MODEL

- ▶ Generates a document containing N words using n-gram
- ▶ A good model assigns higher probability to the word that actually occurs

$$P(W) = P(N) \prod_{i=1}^N P(w_i) \quad p(w) = \prod_{i=1}^N P(w_i) \quad (16)$$

- ▶ The location of the word in the document is not important
- ▶ $P(N)$ is the distribution over N and is same for all documents. Hence it is ignored
- ▶ w_i to be estimated in this model is $P(w_i)$ and it must satisfy $\sum_{i=1}^N P(w_i) = 1$

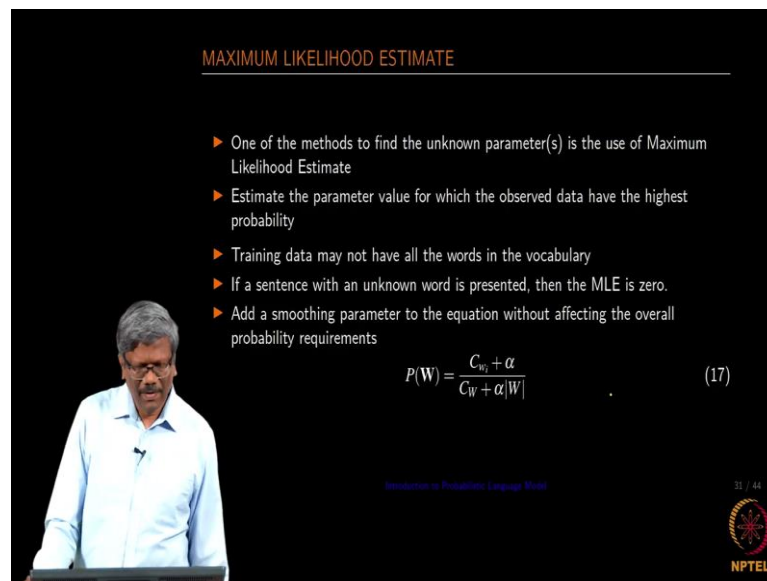
30 / 44
NPTEL

Let us move on to the next as I mentioned earlier we are going to be generating the sentences or going to be predicting the next word. So, let us just put this in the simple mathematical model there should be a smaller one; the probability of a sentence you know the word it is a string of words. So, I am just using the capitals here is nothing but the product of the probability of all the words in that particular sentence and then see this one is the distribution of over N . So, it is going to be common. So, we are going to be

removing this from the formula. So, the formula would be of the form to be like this and then we also want to make sure that the distribution is maintained in this fashion ok.

So, this is a generative model. So, by using this particular formula we are going to be either predicting the next word or say finding the probability of the given sentence or the input sentence ok.

(Refer Slide Time: 08:02)



MAXIMUM LIKELIHOOD ESTIMATE

- ▶ One of the methods to find the unknown parameter(s) is the use of Maximum Likelihood Estimate
- ▶ Estimate the parameter value for which the observed data have the highest probability
- ▶ Training data may not have all the words in the vocabulary
- ▶ If a sentence with an unknown word is presented, then the MLE is zero.
- ▶ Add a smoothing parameter to the equation without affecting the overall probability requirements

$$P(W) = \frac{C_w + \alpha}{C_W + \alpha|W|} \quad (17)$$

31 / 44
NPTEL

And then we also need to understand what is the maximum likelihood estimate among all the words that are going to be coming after two or three words which one has the highest probability. So, that is what we are going to be picking up and that is what we call as the maximum likelihood estimate.

So, this is one of the estimates that we use quite often in the language model. As I mentioned earlier the training data or the corpus that we have used to pick up the unigram bigram and trigrams to generate the language model may not have all the words in the vocabulary. So, I might be trying to form a new sentence with a new word that is not available in the vocabulary. So, if you use the formula the probability of that word is going to be 0; that means, the entire sentence is going to be the probability of the entire sentence going to be 0. So, that is not the right way.

So, you need to probably smoothen this up. So, we are not losing the other elements because of the unknown word that is present in this sentence. So, we just apply some

smoothing parameter. So, the probability of the sentence is not 0 if some words are not present in the given center-right. So, this smoothing parameter will be utilized without really disturbing the probability distribution.

(Refer Slide Time: 09:37)

BIGRAM LANGUAGE MODEL

- ▶ Bigram language model generates a sequence one word at a time, starting with the first word and then generating each succeeding word conditioned on the previous one³
- ▶ A bigram language model is defined as follows:

$$P(\mathbf{W}) = \prod_{i=1}^{n-1} P(w_i | w_{i-1}), \quad (18)$$
 where $\mathbf{W} = w_1, w_2, w_3, \dots, w_n$
- ▶ Estimate the parameter $P(w_i | w_{i-1})$ for all bigrams
- ▶ The parameter estimation does not depend on the location of the word
- ▶ If we consider the sentence as a sequence in time, they are time-invariant MLE picks up the word that is $\frac{n_{w,w'}}{n_{w,o}}$ where $n_{w,w'}$ is the number of times the words w_1, w' occur together and $n_{w,o}$ is the number of times the word w appears in the bigram sequence

³<https://cs.brown.edu/courses/csci1460/assets/files/langmod.pdf>
Introduction to Probabilistic Language Models

32 / 44
NPTEL

Let us now look at the bigram language models; can we really construct a bigram language model or then if we can construct; so, how will it look like ok. So, what going to be doing is I am going to give the theory part of that and then I am going to take you to the real program and then see what and then show you what the model is and then what model parameters are and how are we capturing those model parameters and then what those model parameters contain, how the probabilities for two words the distributed and so and so forthright.

So, now, let us look at the bigram language model. So, we are going to be defining the bigram language model using this formula ok. So, in this case given the word. So, I am going to be finding the next one w_i ; that means, we could give a pair of words if one word is given what could be the possible next word what could be the probability of the next word that could occur after the w_{i-1} word. So, in this case we are going to be estimating the parameters for all the bigrams ok. So, I will give an example so, that you will understand this clearly.

Again this estimation does not depend on the location of the word. So, it could be distributed all over the corpus you know the occurrences of certain pairs of words could

be distributed across the corpus. So, it does not really matter where it occurs whether in the first sentence or in the last sentence or in the middle or anywhere and then to compute this probability we you going to be finding out how many times these two pairs occurred together and then how many times w_i has occurred alone ok. So, that is what we require to really compute the probability of the given bigram word.

(Refer Slide Time: 11:44)

PROBABILISTIC LANGUAGE MODEL - EXAMPLE

1 Peter Piper picked a peck of pickled peppers < / S > (E)

2 A peck of pickled peppers Peter Piper picked

3 If Peter Piper picked a peck of pickled peppers

4 Where s the peck of pickled peppers Peter Piper picked?

—

The joint probability of a sentence formed with n words can be expressed as a product conditional probabilities - we use immediate context and not the entire history

$$P(w_1 | (S)) \times P(w_2 | w_1) \times \dots \times P((E) | w_n)$$

and $P(w_{i+1} | w_i) = \frac{|w_i, w_{i+1}|}{|w_i|}$

—

What is the probability of these sentences?

$P(\text{Peter Piper picked})$ ✓ ✓

$P(\text{Peter Piper picked peppers})$ ✓ ✓

$P(w) = x \dots \times \text{Ⓞ}$

Bigram	Frequency
(S) Peter	1 ✓ ✓ ✓ ✓
Peter Piper	4 ✓ ✓ ✓ ✓
Piper picked	4 ✓ ✓ ✓ ✓
picked a	2
a peck	2
peck of	4
pickled peppers	4
peppers (E)	1
(S) A	1
A peck	1
of pickled	4
peppers Peter	2
...	..
(S) ...	1

So, I am taking here as an example where I have 4 documents ok. So, consider each sentence as a document. So, the first one is Peter , Piper, picked a, peck of pickled peppers and a peck of pickled , peppers , Peter , Piper , picked; If Peter Piper picked a peck of pickled, peppers, where is the peck , of pickled , peppers ,Peter ,Piper ,picked ok.

So, this is the tongue twister; I thought it will be interesting to take this to construct the diagram; bigram that we have the construct the bigram which we will use to construct the language model and then see whether a given sentence is legal in this case or not and so on ok. So, in this case what we do is we first create bigrams, and then for each bigram we compute the frequency. So, for every sentence there is a starting symbol and there is an ending symbol. So, starting a symbol is also considered as a word in this case that is why we see starting symbol and peter as 1.

So, if you look at this a Peter and starting symbol occur once, starting symbol and A occurs once, starting symbol and If occurs 1 and so on. So, this gives an idea of I want to

start a sentence with you know randomly using whatever is available in my model. So, in this case it will find out how many times let us say Peter occurred as the start of the sentence and if Peter occurred several times the probability of picking Peter as the first word in that sentence is pretty high ok.

So, in that way this starting symbol will really help us ok. So, either it is two as you can put it the end symbol here I marked as capital E or it could also be used as slashes as we use in the XML formats ok. So, what I have done is I have computed the frequencies of all those bigrams a Peter Piper; you know how this is computed right.

So, we start with this and then like this and so on. So, every time when we pick a pair of words we find out whether it has occurred earlier or not if occurred we increase the count. So, that is how we have Peter Piper occurring four times you can look at this right. So, this is a second time third time and then here we have the fourth time ok. So, in this fashion all are all these frequencies are computed here.

So, when you want to form a sentence using this corpus assuming that this is what we have and then we want to find out by giving an input sentence which is going to be forming and then find out what is the probability of that sentence given the particular corpus ok. So, finding the probability of this sentence is nothing but the joint probability right. So, you take the probability of each of those occurring words and then multiply that and finally, you will end up with the probability of that given sentence.

So, if you want to find the probability of Peter Piper picked using the bigram and this is your input sentence the Peter Piper is going to be your inputs and Peter Piper picked is going to be the input sentence; compute the probability of this sentence we are going to be looking at the start of this sentence and Peter whether it has occurred; yes, it has occurred once and then Peter Piper has occurred 4 times and Piper picked occur 4 times so; that means, all these three words have been formed in this particular corpus and they have certain frequencies associated with that.

So, there will be a probability that we can compute for the whole sentence without any problem. So, the probability of this particular sentence would be high because we have this sentence, this space occurring here ok. So, let us look at the second one Peter Piper picked peppers ok. So, we have a starting symbol and Peter occurring once right and then Peter Piper we know it is occurring Piper picked; yes, we have it and then picked

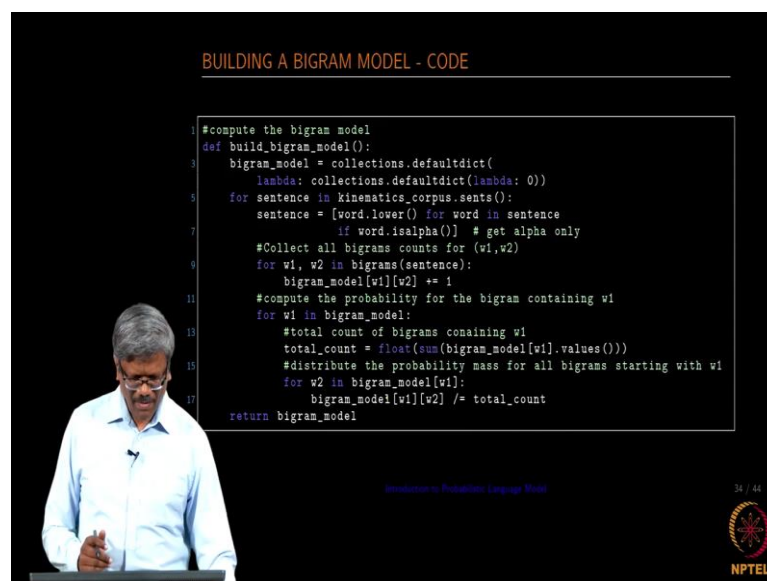
peppers. So, do have pickled peppers on this; we do not have this; right. So, what is going to be the probability of this right?

So, this particular one occurs. So, many times we are going to be looking at pickled peppers the count of pickled peppers which is 0; that means, in the probability when we look at it. So, we are going to be having 0 by the number of times pickled occurred in the entire corpus; that means the entire probability of this sentence is going to be 0; I think which is not correct; is it not it? So, some of the sentences, some of the words have occurred but only because of this word which is not coming as a pair here we are going to have a problem.

So, what we do here is. So, we are going to be using some kind of smoothing parameter. So, this particular 0 is going to have some value other than 0 is greater than 0. So; so, the probability of this sentence will not be as high as what we see here and but we will have some value related to that.

So, in long paragraphs and sentences and if you want to compute the probability of whether the paragraph is legal within the context of given corpus. So, we cannot have this approach because the entire paragraph then becomes 0 in that case; correct. This when we want to compute the probability of a sentence we would use a formula of this type so that none of the probability of the bigrams would be equal to 0; ok alright.

(Refer Slide Time: 18:22)



BUILDING A BIGRAM MODEL - CODE

```
#compute the bigram model
def build_bigram_model():
    bigram_model = collections.defaultdict(
        lambda: collections.defaultdict(lambda: 0))
    for sentence in kinematics_corpus.sents():
        sentence = [word.lower() for word in sentence
                    if word.isalpha()] # get alpha only
        #Collect all bigrams counts for (w1,w2)
        for w1, w2 in bigrams(sentence):
            bigram_model[w1][w2] += 1
    #compute the probability for the bigram containing w1
    for w1 in bigram_model:
        #total count of bigrams containing w1
        total_count = float(sum(bigram_model[w1].values()))
        #distribute the probability mass for all bigrams starting with w1
        for w2 in bigram_model[w1]:
            bigram_model[w1][w2] /= total_count
    return bigram_model
```

34 / 44

NPTEL

So, now, you know we have to look at this in action. So, far we have only been seeing it with respect to the theoretical formulation and so on.