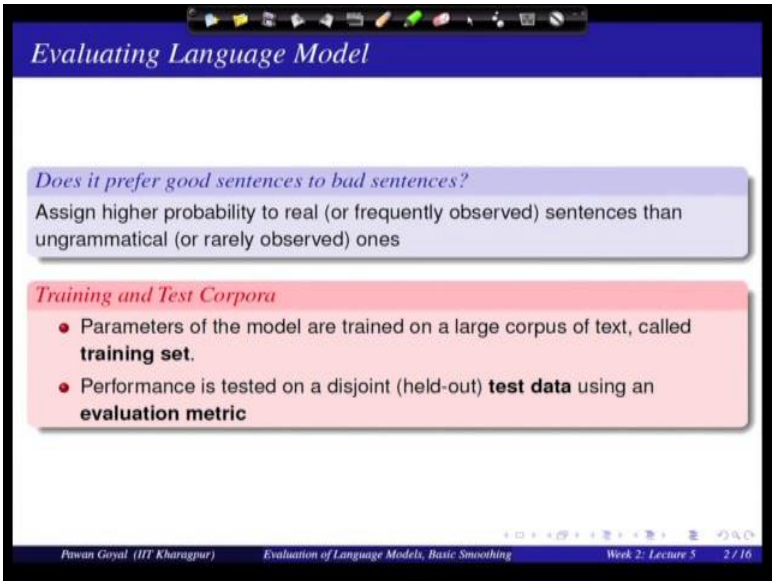
**Natural Language Processing**
**Prof. Pawan Goyal**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Kharagpur**

**Lecture - 10**
**Evaluation of Language Models, Basic Smoothing**

So, welcome back for the final lecture for second week. So, in the last lecture, we were discussing about the basic of language modeling; and there we just hinted about one problem about the zeros in language models. So, today in this lecture, we will discuss not only how to handle this zeros in language modeling's, but also how do you go about evaluating your language model that you have learned from some data. So, these are the two main topics for this lecture.

(Refer Slide Time: 00:49)



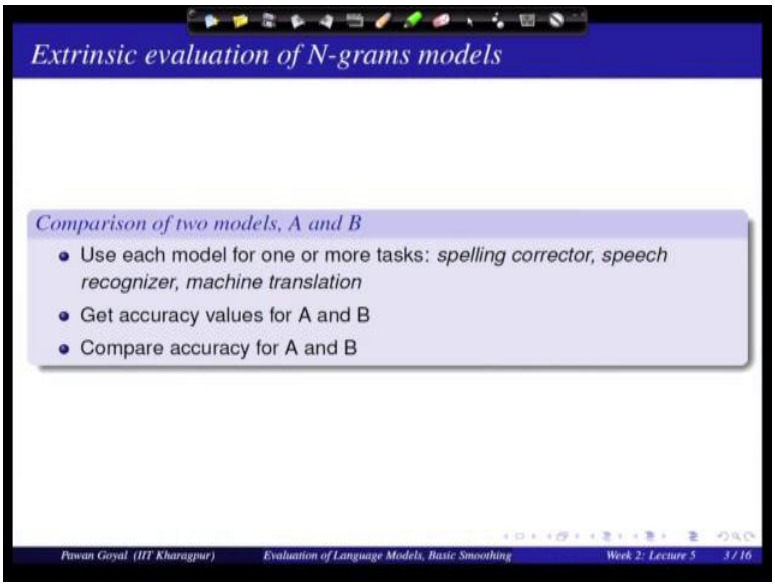

So, we start with the evaluation of language models. So, what is the basic criteria for evaluating a language model? So, one symbol intuition could be a language model is better if it is assigning a high probability to the actual sentences, the real sentences and a low probability sentences that are not so grammatical that can be one criteria for evaluating language models. But this has to be done manually. So, you will have to check what is a probability that your model is assigning to various sentences that are real, most of the sentences that are not real that are not grammatical, but can we do that

in an automatic manner. So, for that we may also use some sort of training and test data. So, what is the idea behind using this training and test data?

So, I have some corpus and I learn my language model on some part of that. So, this is my training set. Now, once I have learned that language model from my training set, I will test whether it is providing a good high probability for my test data. And I will have some evaluation metric for finding out how good this model is doing on my test data, and by this method I can even compare across different language models. So, if I have learned three different language models for example, I can find out which one does better on my test data, and that will be the best model. So, in general what are the different ways in which language models can be evaluated. So, as such there are two different criteria for evaluation, one is called extrinsic, another one is called intrinsic. We will discuss what are these individually.

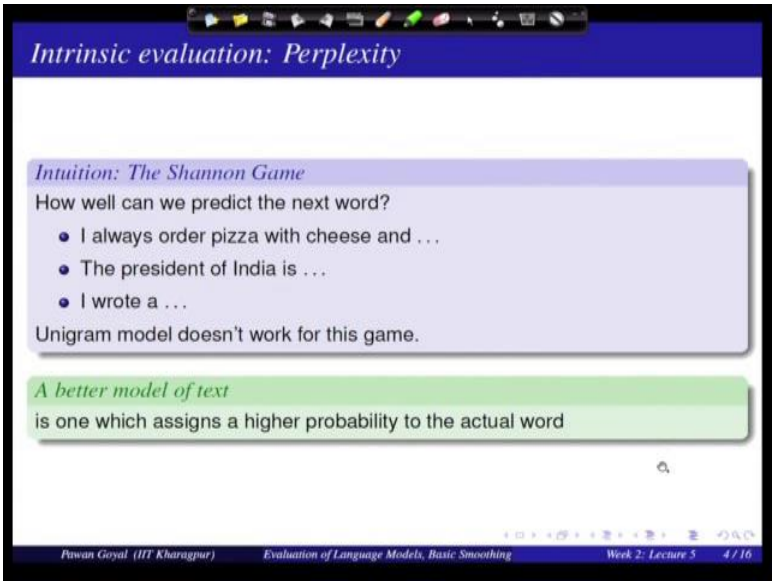
So, what do I mean by extrinsic evaluation of my language models? So, suppose I have learned two different language models a and b from some training data. Now, what do we do in extrinsic evaluation, we try to use them on certain task. So, remember some of the applications that we have discussed of language models, so they can use in the spelling correction or in speech recognition and all that. So, now, what I will do in this extrinsic evaluation is that once I have learnt two different models, I will try apply them to these tasks individually, and then see what is the performance that I am obtaining for each of this task by different models. Now, I will say whichever model is giving a better performance on these task is my preferable model. So, this is called extrinsic evaluation or task based evaluation.

(Refer Slide Time: 03:32)




So, here we have discussed if we have mentioned three different tasks a spelling correction, speech recognition and machine translation. I will have two models A and B, I will get accuracy values for these three tasks and I will compare the accuracy and tell which language model is preferable than other. So, this is my extrinsic evaluation but is there some intrinsic way of evaluation without actually applying it on some task.

(Refer Slide Time: 03:57)

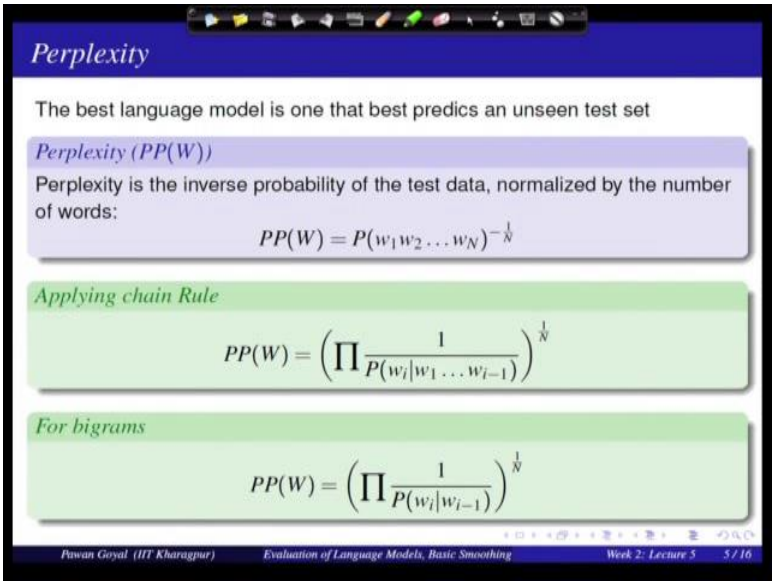



So, for that so we have the notion of perplexity, so this is how we evaluate language model for intrinsic evaluation. So, intuition comes from the Shannon game if you heard

about this name. So, what is that game how well can you predict the next word given certain context. So, let us see some examples. So, I have the first sentences here, I always order pizza with cheese and there is a word that you have to predict may be you can say pepper or whatever. And similarly, the second sentence, the president of India is and you can predict the next word. Similarly, in the third sentence, I wrote a and you can predict letter or program or whatever.

So, now in my data I know what are the words that will fill up these blanks. Now, suppose I have two different language models, I will find out which of these fills up these blanks better than the other one. So, whichever one is good at predicting the next word is my preferable language models. So, one thing you can easily say from here, the unigram models are probably not built for this task. So, suppose you want to predict the next word using a unigram model, what would happen? So, if you remember unigram models do not use the context at all. So, you will just end up providing at every place the word that is having the highest probability. So, probably unigram models are not good for this task, but you can apply a bigram models the model which uses the previous word, trigram models that uses the previous two words and so on. So, finally, among different language models a better model is one that you will assign a higher probability to the actual word.

(Refer Slide Time: 05:54)

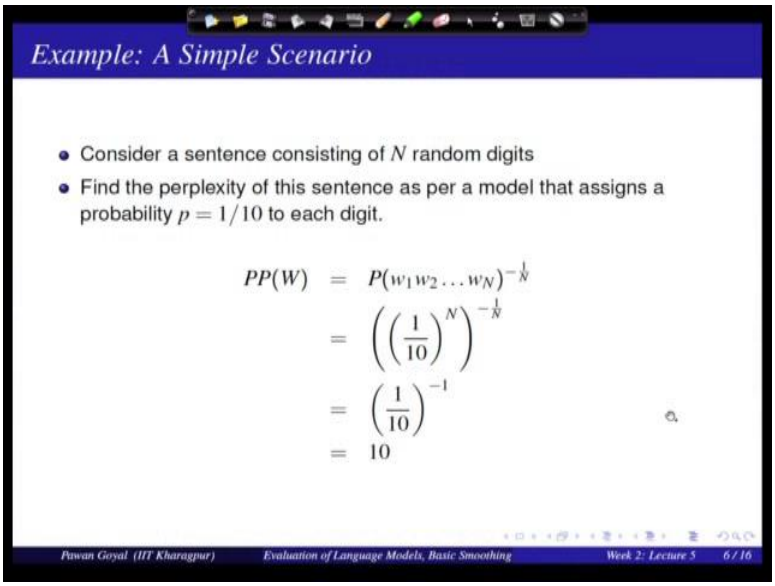


Now, what is the formal definition of perplexity? So, in general, what we are trying to find out the best language model that predicts an unseen test data. So, how do I define perplexity? So, this is a simple definition of perplexity. I have some test data; I am finding out the inverse probability of test data normalized by the number of words that I am seeing in my test data, that means finding the probability of test data and then normalize it with respect to the number of words and its inverse probability that means, if I have a low perplexity I have a better model. So, formally I can define my perplexity like that suppose in my test data I have words w 1 to w N, I am finding the probability of this whole sequence and normalizing it by minus 1 by N here.

So, now this is a general definition. Now, the next question that you might have is that where does the language model come into picture in this definition. The simple definition of finding probability of this whole address w 1 to w N and sub with some normalization. Now, where does the language model come into picture? Now, remember the chain rule that we discussed of probabilities if I have this sequence w 1 to w N, how can write the probability of the sequence in terms using the chain rule of probability, so that is what you will see here. So, I can write the same expression like that one divided by probability of w i given the previous i minus 1 words this is in general this definition of probability w 1 to w N.

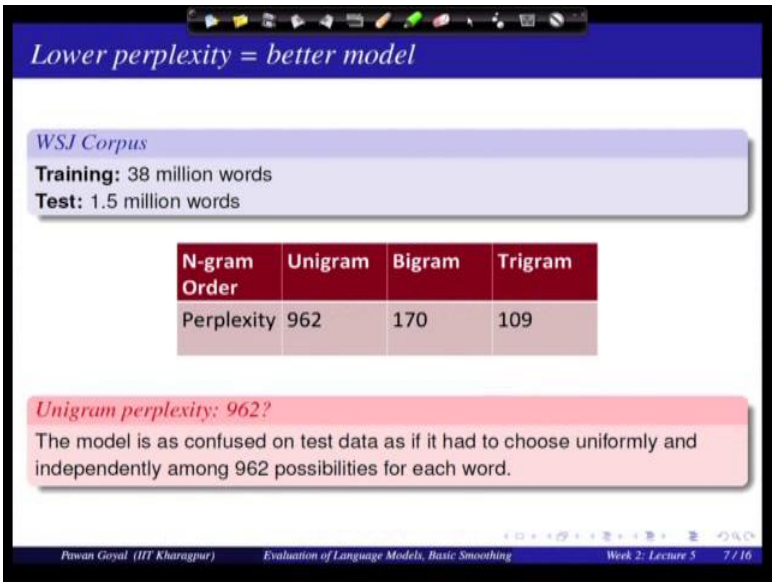
Now, can you see how do we apply language models in this definition. So, in language model, we take one particular assumption about using this chain rule that is how many previous context I will be using. So, you can in place of this, you can replace any of the model that you have learned. So, suppose you want to replace with your bigram model that means, you want to find out what is the probability that the bigram model will give for this utterance. So, what you will do? So, we simply replace here the bigram model probability, so that will give you the perplexity for the bigram model that you have learned or trained using some training data. So, you will feed in all this probability w i given w i minus 1 in that model and that will give you the perplexity of the bigram model. So, now just to give you an intuition what do I mean; what the number of number that I will get indicates perplexity value what does that indicate, let us take a simple example.

(Refer Slide Time: 08:33)




So, suppose I have a sentence that contains N random digits. And I have a model that assigns a probability of 1 by 10 to each digit. Now, I want to find a perplexity of the sentence using that model that will give me a probability of 1 by 10 to each digit. So, what will be the perplexity, let us try to use the definition. So, how did we define perplexity? That is probability of this whole sequence yes, w 1, w N to the power minus 1 by N, yes. So, can you try to fill in the values? Suppose my model gives a probability of 1 by 10 to each word. So, this would be 1 by 10 to each word to the power N, for N different words N to the power minus 1 by N, so that is nothing, but 1 by 10 to the power minus 1 and that will give me 10. So, this tells me that the perplexity of this model is 10, the model that assigns a probability of 1 by 10 to each digit. Now, this might give you a hint of what this number indicates. So, let us take another example from some test data. So, what kind of perplexity values we get and how we can interpret this values.

(Refer Slide Time: 09:57)

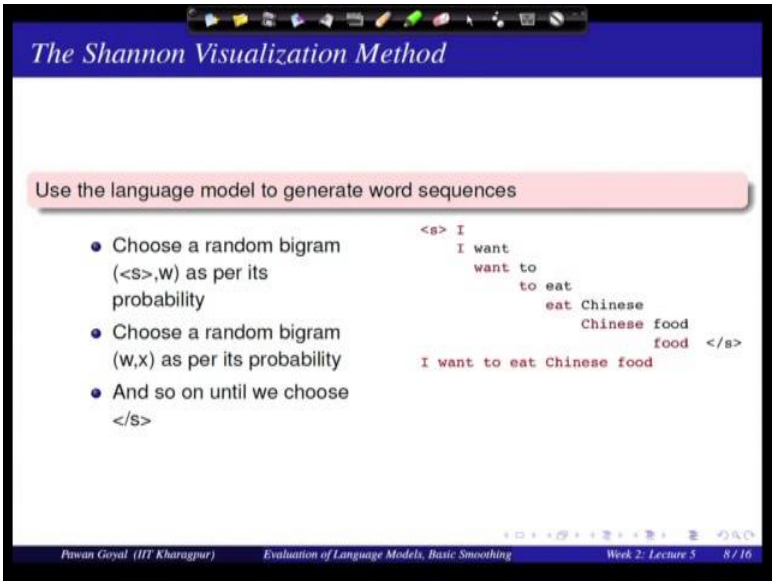



So, an experiment was conducted over wall street journal corpus. So, in training they had 38 million words on which the language model was trained. And for testing they had 1.5 million words on which perplexity was computed. And they trained three different models unigram model, bigram model and trigram model. Now, and they found out what is the perplexity of the model in using the test set. So, these are the numbers that were found; for unigram model perplexity was 962, bigram 170, and trigram 109. So, now let us try to answer this question, what is this value of 962 perplexity in unigram indicate. So, what it means is that whenever my model is trying to assign a word, as if it has to choose among 962 different possibilities at each individual choice point independently and randomly.

So, this means the model is very, very perplexed. So, if the perplexity is high my model is very, very confused; if it is low my model is not so much confused. So, in this case, what you are seeing if I use a unigram model perplexity is 962 the model is very, very perplexed, but if I go for a bigram model, it is 170 the model is not so much perplexed now. Trigram model it becomes even better it is only 109, so that is what you are seeing unigram per perplexity of 962 means the model is as confused on the test data as if it had it had to choose uniformly and independently among 962 different possibilities for each word. And that you can also relate with your previous example, because every time it had it had to choose among 10 different possibilities for each digit because it is about

giving a probability of 1 by 10 to each digit. So, now, you understood what the perplexity means.

(Refer Slide Time: 11:58)




So, once we have built a language model we can also use it for other tasks. So, one very interesting task is called Shannon visualization method that is can I use this language model to visualize or generate sentences. So, if you have read the original paper of Shannon on the mathematical information theory, so there he uses this method to generate various sequences of words. So, what is the idea? Suppose I have learnt a language model can I use that to generate various sequences of words or sentences. So, how do we actually apply this Shannon visualization method? So, suppose I have to generate a sentence, so what I will do, I will have to generate a sentence. So, I will first choose a random bigram that is starting the sentence as per the probability. So, what do I mean by that.

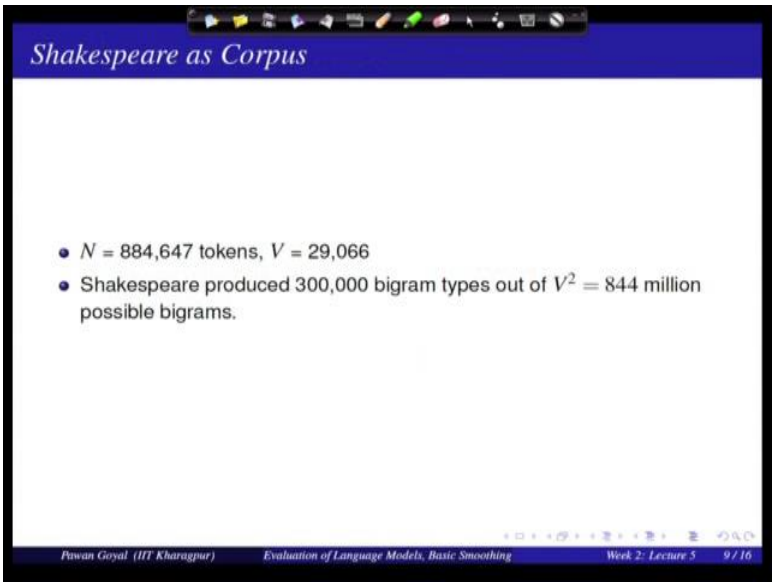
So, I am learning my bigrams. So, I have in with the start of the sentence whether the word w 1 occurs with what probability. With the start of this sentence, word w 2 occurs with some probability. So, this is a complete probability distribution that will add up to 1. Now, if I have to generate a sentence, I have to choose one among this possibilities and this will depend on the probability of that bigram. So, this you can think of as multinomial distribution and you are sampling one word from this multinomial

distribution. So, suppose you pick picked up a particular word here. So, you start generating a sentence you start and w 2.

Now I have to choose the next word. So, that is the next step choose a random bigram as per its probability. So, what I will do now I will go to the distribution w to n different words w 1, w 2 and so on. And from this distribution again from this multinomial distribution I will sample one sample one word suppose I find sum w 50. Again I will try to a sample word with its distribution w 50 and the next word. Now, the question is when do I stop when do I say that my sentence is complete, I will say my sentence is complete once I sample a word with the end of the sentence. At some point I sample w i and the next word is end of the sentence I say my sentence is closed. So, this is the whole idea of Shannon visualization method.

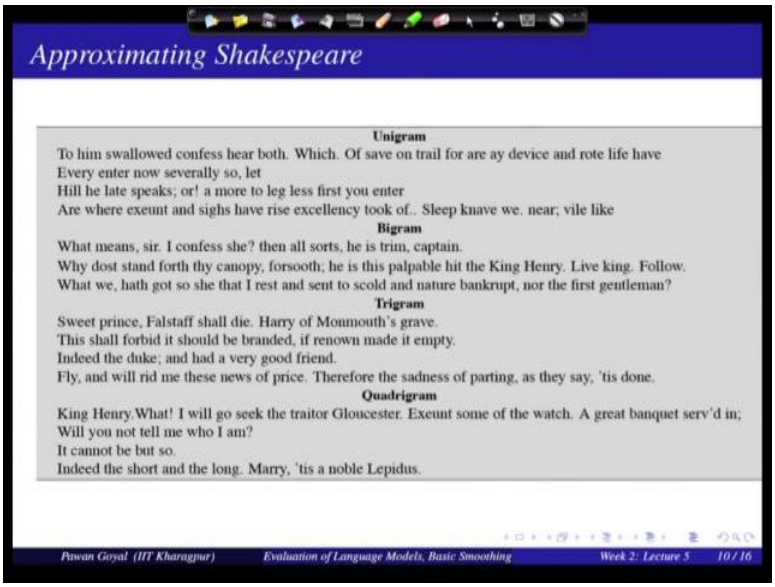
So, I do that until I choose end of the sentence and we will see one example from one corpus. So, from restaurant corpus that we discussed in the last lecture, suppose I have learned my bigram model and I want to generate a sentence. So, how will I do that I choose start of a sentence find out the first word, I find i, then I take the first word as I choose the next word how do I choose the words by sampling from the multinomial distributions. So, in this case, if we sample we might end up with getting the sentence I want to eat Chinese food and after food we get the end of the sentence. So, I want to eat Chinese food would be a sentence that is generated by this method.

(Refer Slide Time: 15:36)



Now, suppose we try to use this method over the Shakespeare's corpus. So, we have some number of tokens and the vocabulary size is 29,066. Just to give you an indication that bigrams are actually very, very sparse. So, if you take the vocabulary size of 29,000 something how many bigrams are possible. So, what do you mean by a bigram - two words together. So, I can take say V square are the possible bigrams because every possible combination can occur, but so this gives me a number of 844 million possible bigrams, but in the corpus how many bigrams were actually observed. So, we find there were only 3,00,000 bigrams that we have observed in the corpus. So, this is very, very sparse. So, now, suppose I build various language models from this corpus and try to generate various paragraphs and sentences. So, what do we actually observe, you will see what happens if I take unigram model, bigram model, trigram model and high order models.

(Refer Slide Time: 16:43)



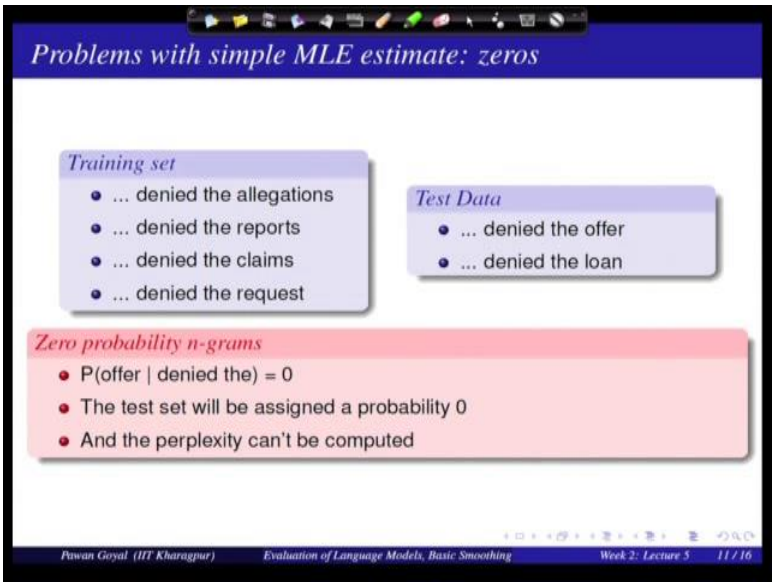

So, here you are seeing, if you take unigram model, you are getting some words that are probably very popular in Shakespeare's, but the sentence themselves are not making sense. So, you see the word like which here, it occurs independent of any other words and this becomes a complete sentence that will not probably something that you will see in the Shakespeare's corpus. So, now suppose you go to the bigram model now. So, you start making some sense. So, you have words sequences like what means, I confess and all sorts, they have good bigrams, but if you try to observe the sentence probably they are not making much sense.

Now, if you go to trigram model then again you have getting some more sense falstaff shall die and the shell forbid, it should be, should be brand, they have some nice sequences again and this is starts making a much more sense in terms of sentences. And if you go to quadrigram, then you are getting something that is resembling King Henry, What I shall go see the traitor Gloucester. So, this look like a valid sentences from the Shakespeare's corpus. So, that is the idea as you go to higher and higher order model, the kind of sentences you generate will be something that actually resemble the corpus from which we are training this language model. So, this is my visualization method.

So, now we will try to go to the problem of a smoothing that we discussed in the last lecture. So, remember what the problem was in my language model suppose I am training for bigrams, so yes. So, just take the statistics that we saw from Shakespeare's corpus. There were 844 million possible bigrams out of which only 3,00,000 bigram actually occur. So, now if you assign the probability to each bigram very few will get a probability greater than 0, others will get a probability of 0. Now, suppose you are taking a test data and finding out how much it resembles Shakespeare's corpus. And whenever you see a bigram you are taking probability from the trained language model. Now, suppose a bigram occur that is not there in the Shakespeare's corpus, imagine the probability is 0.

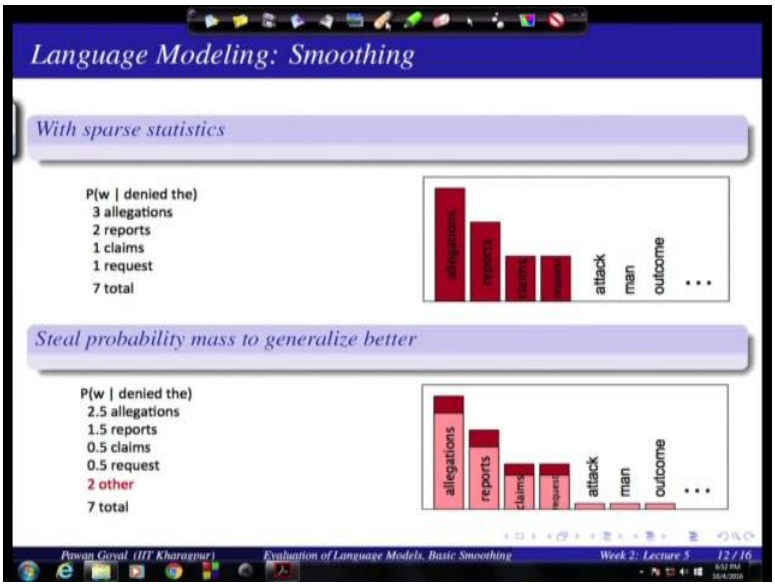
Now, what happens to perplexity value remember this is simply the multiplication of all the different probabilities. So, this will becomes 0, so that will not to very, very helpful. So, you actually would like to give it some probability, so that this does not become zero that is why we will study topic of smoothing, how we can assign different probability values to get around this problem of 0s.

(Refer Slide Time: 19:26)




So, let us take a simple example. Suppose I am learning my trigrams and my training data I have seen the following sentences. I have seen these many occurrences denied the allegations, denied the reports, denied the claims, denied the request and I am learning a model of finding out the word after denied the. Now, suppose in my test data, I see these two occurrences, denied the offer and denied the loan. So, what would happen to the perplexity of my model? So, in the test data, I have seen these two occurrences that were not there in the training data. So, probability of offer given denied the will be 0, same with probability of loan given denied the. So, the test set will be assigned a probability of 0 and the perplexity cannot be defined, and that is what we were saying initially. So, how can I go around this problem? So that I can compute my perplexity even if there are certain bigrams or trigrams they did not occur in my training data. So, what is the idea of smoothing?

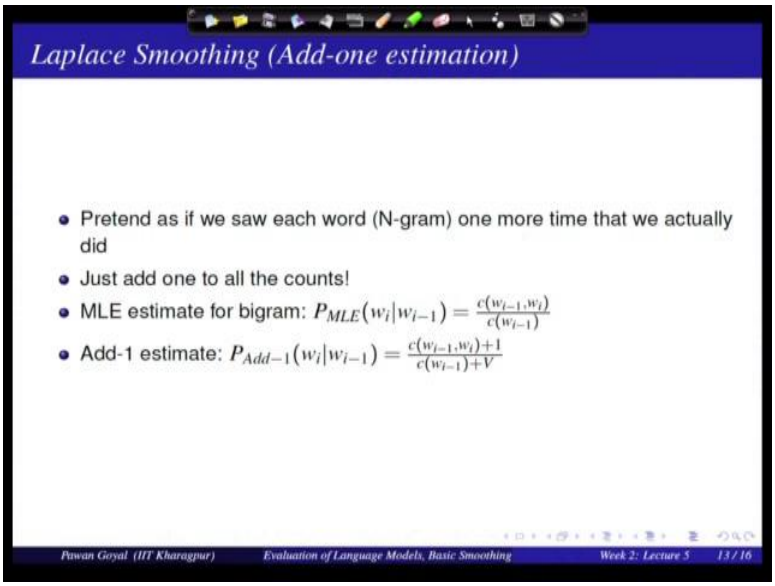
(Refer Slide Time: 20:38)




So, idea is suppose so this is what we were seeing. I am computing the trigram model for the word w after the occurrence denied the. And suppose in my data I see four different words right, I see allegations, reports, claims and request in total seven words. And I can assign the probability to each of these as 3 by 7, 2 by 7, 1 by 7, and 1 by 7 and that is what you are seeing in this plot 3 by 7, 2 by 7, 1 by 7, 1 by 7. So, these adds up to 1. Now, what about the probabilities for the other trigrams like denied the attack, denied the man, denied the outcome, these are also very, very feasible trigrams. So, here what will happen we will assign a probability of 0 to all of these yes. So, all these have a probability of 0. Now, what is the idea of smoothing? The idea of a smoothing is can I take some probability mass from each of these four words and assign that to the three words or any other words that I have not seen in my training data.

Can I steal some probability mass from these four words to assign some probability mass to the other words in my data? So, that is suppose here I have taken a probability mass of 0.5 each from the four words so that means, I get a probability mass of 2 by 7, and that mass I distribute among all the other words in my corpus. And this can be distributed in multiple different ways, we will see some possible ways in which we can distribute this stolen mass to the other words that we were dint see in my training data and how much mass has to be distributed that also we will see. So, there are different methods that do that. So, the basic idea is clear. So, how exactly we do that?

(Refer Slide Time: 23:09)



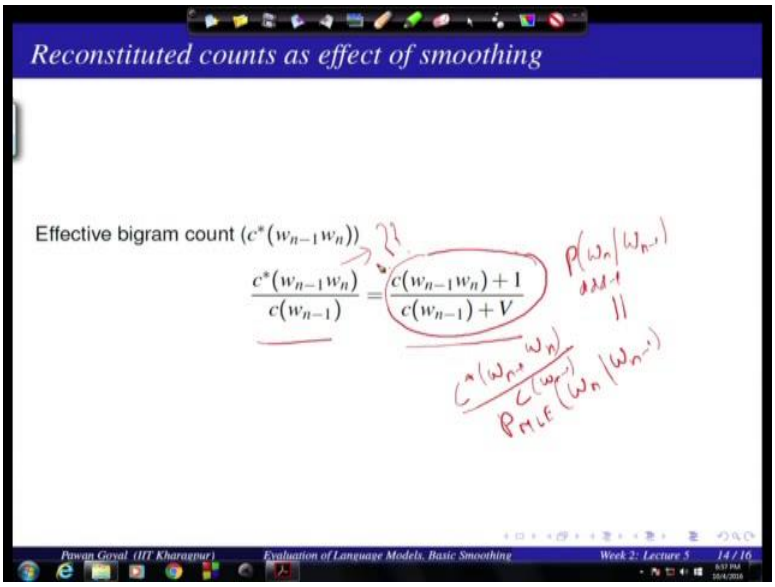

So, a simple method is called Laplace smoothing or add one estimation. So, what is the idea? So, pretend as if you have seen every n-gram one more time that we actually did in my training data, so that is suppose I saw it only once, so I will pretend as if I have seen it twice. So, what will happen to the n-grams that I have not seen at all in my training data? So I will pretend as if I have seen them once, so this is simple idea. So, we will just add one to their actually counts that we found from the training data, we will just add one to that. So, remember this maximum likelihood estimate for this bigram probability of w i given w i minus 1, we find it using number of times if we observe w i minus 1 followed by w i divide by the number of times I observe w i minus 1 in my training data. So, this is my definition of MLE. So, how do I estimate language model or a bigram model using MLE?

Now, how do I use the Laplace's smoothing there? So, what did we say? We will pretend as if we have seen each n-gram one more time than we actually did. So, what we will do here. So, we will add 1 to the actual count. So, will do that for each possible bigram, but to ensure that the probability adds up to 1 I have to make some modifications to my denominator, so that they are normalized. So, what will I add to my denominator?

So, to get this answer, you can see how many different bigrams will be there for which I will be adding 1. So, how many different w i's will be there that will be number of words in my vocabulary; that means, I will add 1 capital V times. So, to normalize, I will also

have to add a V here in the denominator and that is essentially the idea of my add one smoothing that I add a 1 to my actual n-gram count and add a V in my denominator to normalize it. So, this is my add one estimate number of counts plus 1 divide by the count of the unigram w i minus 1 plus the vocabulary size, so that is the very, very simple smoothing technique that you can use in general. So, that does not require a lot of fancy estimates, you can just get your language model and easily apply this smoothing method.

(Refer Slide Time: 25:54)

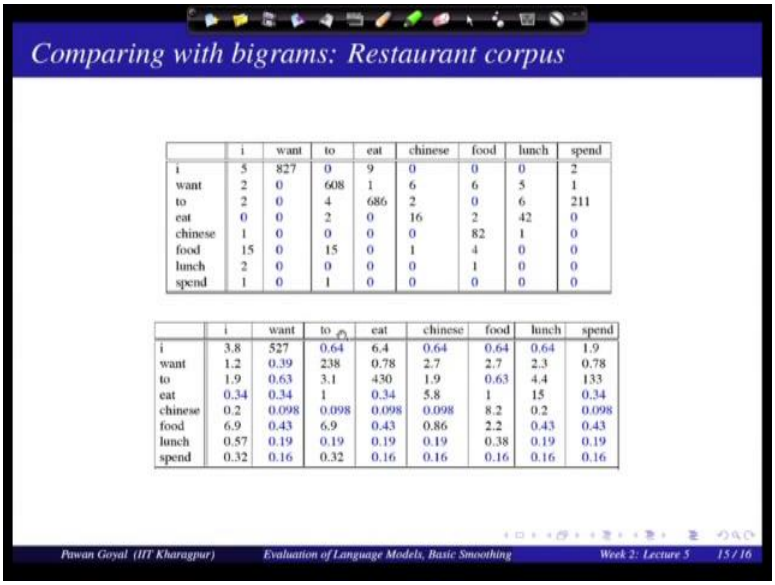



Now, when we apply this add one smoothing method we can also talk about what is the effective bigram count. So, let me understand what do I mean by this effective bigram count. So, what I mean is, so you see here you have modified your actual counts or the probability, what is the idea? So, now, what is in effective what would have been the count in your actual training data such that you have got the same would have got the same probability. So, what I am saying. So, this is your new probability, probability w n given w n minus 1 as per the add 1 smoothing, yes.

So, question here is what would have been my effective bigram count c star w n minus one w n such that if I do the MLE estimate PMLE, w n given w n minus 1, I will get the same value as this. So, how do I get the effective bigram counts? So, this effective bigram count if I would divide by counter w n minus 1, I should have got this probability, so that is means I can compare c this probability with this probability and that will give me what is my effective bigram count. So, that is how I define my effective

bigram count. So, we will see some example. So, if I apply this for my restaurant corpus, so what kind of effective bigrams do we observe.

(Refer Slide Time: 27:43)




| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 5 | 827 | 0 | 9 | 0 | 0 | 0 | 2 |
| want | 2 | 0 | 608 | 1 | 6 | 6 | 5 | 1 |
| to | 2 | 0 | 4 | 686 | 2 | 0 | 6 | 211 |
| eat | 0 | 0 | 2 | 0 | 16 | 2 | 42 | 0 |
| chinese | 1 | 0 | 0 | 0 | 0 | 82 | 1 | 0 |
| food | 15 | 0 | 15 | 0 | 1 | 4 | 0 | 0 |
| lunch | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| spend | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

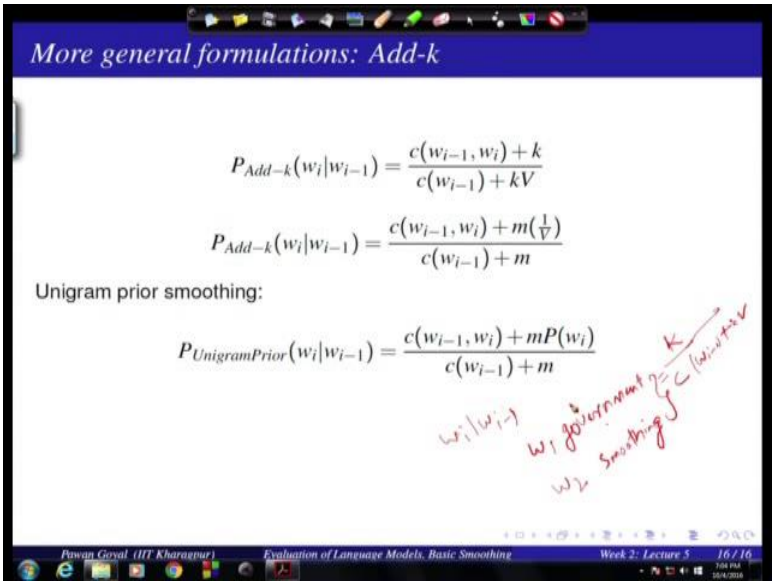
| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 3.8 | 527 | 0.64 | 6.4 | 0.64 | 0.64 | 0.64 | 1.9 |
| want | 1.2 | 0.39 | 238 | 0.78 | 2.7 | 2.7 | 2.3 | 0.78 |
| to | 1.9 | 0.63 | 3.1 | 430 | 1.9 | 0.63 | 4.4 | 133 |
| eat | 0.34 | 0.34 | 1 | 0.34 | 5.8 | 1 | 15 | 0.34 |
| chinese | 0.2 | 0.098 | 0.098 | 0.098 | 0.098 | 8.2 | 0.2 | 0.098 |
| food | 6.9 | 0.43 | 6.9 | 0.43 | 0.86 | 2.2 | 0.43 | 0.43 |
| lunch | 0.57 | 0.19 | 0.19 | 0.19 | 0.19 | 0.38 | 0.19 | 0.19 |
| spend | 0.32 | 0.16 | 0.32 | 0.16 | 0.16 | 0.16 | 0.16 | 0.16 |

So, we remember this is what this is the counts that we see in my bigram, this my restaurant corpus. So, we saw that in the last lecture, I want occurs very high number of time sense and so on. So, this is the actual bigram that we see. Now, suppose I apply add one smoothing, so you cannot apply to this data right exactly because is not the complete data this is just a small screenshot. So, now suppose you use add one smoothing and then if you find that what is my effective bigram counts. So, what kind of effective bigram count do you see? You see in this table certain counts are 0. So, one thing you would assume that after applying this smoothing technique, these will not be 0; this will be greater than 0. And whatever for having a high value should have a smaller value because some mass would be stolen from there to give it to the words they did not occur and that is what exactly you will see when we apply this add one smoothing and then do the effective bigram counts. So, these are my effective bigram counts.

So, you see here whenever the word, I want the bigram I want occur 827 times initially the effective bigram count now is only 527. On the other hand, I too did not occur at all in my training data, but now it has effective bigram count of 0.64. Another observation you can make from here given the previous word for the next word if that occur 0 times the effective bigram count remains the same, so its 0.64 whenever the previous word is I

for the words like to Chinese food and lunch, but this varies across different previous words. So, if you take the word like want, for wants this value becomes 0.3 time for to it becomes 0.63. So, this depends on the previous word. And that you can also see why because I am doing this plus v to the count of the unigram, yes and that will be different for different words that is why this value will be also different for different words.

(Refer Slide Time: 29:48)




So, in general, so here we talked about the add one estimation, but in general there are some simple variants of this also. So, one simple variant is called add-k estimation. So, we are adding one to each bigram. So, why want one, why cannot we do some general thing like k, k can be 0.5 or more than 1 depending on how big your data is. So, this is called add-k estimation. So, here what is the idea, we add-k to each count and accordingly we will add-kV to my denominator, yes. So, this is actually the same as add one estimation if I take k is equal to 1.

Now, I can also make some variant here. So, suppose I say KV is equal to m. So, now, I can write it as this plus m, and this plus m by V, yes, this is effective the same, now m is k times v. So, this is another variant. And this also gives me an idea on how I can improve this basic smoothing method. So, here what I am doing, I am adding m by V to each word effectively I am doing m times 1 by V to all the V words in my vocabulary. And if we add up for all the words in a vocabulary m by V, you get an m. So, now, can we do something better there the idea is instead of adding m times a uniform one by V to

each word can you add m times the unigram probability for the word. And you say this will add up to one for all the words you will effectively end up getting the same values in numerator and denominator when you normalize the probability, but this might be a better estimate, why is that.

So, let us first see all these variations. So, this is when I replace kV by m that is what I get. So, here in place of 1 by V, I can also replace the probability of the word that will be a different smoothing, but what we are saying this might be a better way of doing a smoothing this called unigram prior smoothing. So, let us just discuss this point why this might be a better way of doing a smoothing. So, remember what we are trying to do here, we are trying to find a probability or different words that do not occur in my training data w i given w i minus 1. Take a word w i that is very, very common. So, like I have a word government that is very, very common and I might have some other word like may be something like smoothing that may be other word w 1, w 2.

What are you doing in your add one smoothing or add-k smoothing? You are just studying k or one and dividing by the count of the previous word plus kV and this will be same for both the words, yes. So, what is the idea of unigram prior smoothing? Idea is that if I know that this word is more common in my corpus then this word probably I can say that the probability of this word occurring of a w i minus 1 will also be higher than the probability of this word occurring after w i minus 1, and that is I am trying to exploit. This data I have from my corpus and I am trying to exploit that for doing my smoothing. So, this is called unigram prior smoothing. So, in general so there are many other ways of doing smoothing also.

So, we discussed add one smoothing and the institution behind unigram prior smoothing, but there are other advance models of smoothing also that we will see in the next lecture, so in week 3.