

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

Lecture – 11
Language Modeling: Advanced Smoothing Models

So, welcome back for the third week of this course. So, in the last week, where we ended we were discussing about language modeling, we talked about the basis of language modeling then we talked about what how to be evaluate a language model and then we came to the topic of smoothing. So, what was the simple idea we discussed, what is my advanced smoothing where I add one to each individual count, and accordingly I do some normalization in my denominator. And we added by saying there are certain simple variance possible like at K , I can also do the unigram prior smoothing by instead of adding uniform weight to each word, at a weight that is depending on the probability of that word unigram probability for that word. And that we were arguing that might work better than simply adding uniform count to each of the bigrams.

There we had certain parameters like what will be a best way of choosing K and M in different models. So, simple answer would be, so there is no unique value that you can choose. Suppose you have some held out data, so we can try to find out for which value of K and M , you are getting the best publicity for that held out data and that can be one possible way of choosing the values of K and M . So, now today, we will talk we will go beyond the advanced smoothing and talk about some other advanced smoothing methods.

(Refer Slide Time: 01:52)

Advanced smoothing algorithms

Some Examples

- Good-Turing
- Kneser-Ney

Good-Turing: Basic Intuition

Use the count of things we have seen once

- to help estimate the count of things we have never seen

Pawan Goyal (IIT Kharagpur) Language Modelling: Advanced Smoothing Model Week 3: Lecture 1 2 / 18

So, we will start with two different smoothing methods called Good-Turing smoothing and Kneser-Ney smoothing. So, what is the intuition behind using a Good-Turing smoothing? So, intuition is essentially I want to find out what should be the probability for the words that I have not seen in my training data. So, what Good-Turing is smoothing says; so why do not you use the estimate of the things you have seen once in your training data you estimate about the things we have not seen. Similarly, whatever you have seen twice use that estimate that things you will see once and like that.

So, this is just to adjust the probability mass, so that the things that were seen twice are used to find the probability for the things that were seen once. Things that were seen once are used to estimate the probability for the things that were not seen at all. So, the idea is that you have to use the count of things we have seen once to find the count of things we have never seen.

(Refer Slide Time: 03:02)

N_c : Frequency of frequency c

Example Sentences

<s>I am here </s>
<s>who am I </s>
<s>I would like </s>

Computing N_c

I	3	
am	2	
here	1	$N_1 = 4$
who	1	$N_2 = 1$
would	1	$N_3 = 1$
like	1	

Pawan Goyal (IIT Kharagpur) Language Modelling: Advanced Smoothing Model Week 3: Lecture 1 3 / 18

So, for that let us first define what is the frequency of frequency. We used this one of the earlier lectures also. So, we take this three sentences, I am here, who am, I would like. From these sentences, I am trying to construct the frequency of frequency, so that is how many words occur ones, how many word occurs twice and so on. So, if you see here, the word I occurs thrice, am occurs twice, and the other four words occurs once. So, what will my frequency of frequency? I will find out how many of words occurs ones with frequency one. So, that is my N_1 . So, there are four words here that occur only once. So, N_1 is 4. Now, you will see how many words occur twice, so how many words have a frequency of 2, so that will I will have 1 here N_2 is 1. So, this is my frequency of frequency. So, four words occur once, one word occurs twice, one word occurs thrice.

(Refer Slide Time: 04:07)

Good Turing Estimation

Idea

- Reallocate the probability mass of n -grams that occur $r+1$ times in the training data to the n -grams that occur r times
- In particular, reallocate the probability mass of n -grams that were seen once to the n -grams that were never seen

Adjusted count

For each count c , an adjusted count c^* is computed as:

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

where N_c is the number of n -grams seen exactly c times

Handwritten notes: N_{c+1} is count, N_c is count, N_{c+1} is count

Now, how do I use that in Good-Turing estimate? So, what is the idea? So, this we discussed earlier. So, I want to reallocate the probability mass of n -grams that occur r plus 1 times in the training data to the n -grams that occur r times, especially you want to see the n -grams that occur once to find the probability mass for the n -grams that occur 0 times, but you that you do that in general. So, the n -grams that occur r plus 1 times used that for r times.

So, now in particular, reallocate the probability mass for n -grams that were seen once for those that were never seen. So, now, we should see formally what will be the effective probability mass N count that we will get by this Good-Turing estimate. So, let us see suppose I want to find out four words that occurred c times, words that occurs c times what will be the effective adjusted count after applying Good-Turing estimation, now how we defined that? So, what would happen? In my training data, I will have words that occur zero times words that occur one time let it be N_0 , N_1 words that occur twice N_2 and so on. So, N_2 numbers that occur twice, similarly now there are N_c number of words that occurs c times N_{c+1} words that occurs c plus 1 times.

Now, I want to find I want to use Good-Turing to estimate the count for these words that occur c times. So, what it be say we said we will use this probability mass and give it to these words. Now how to we distribute. So, what is the probability mass associated with the words that occurs c plus 1 time. So, now suppose there are N words or n th bigrams in

total. So, now what is the probability mass for the bigrams that occurred or n-gram we can say in general that occurred c plus 1 times, you see how many n-grams are there are N_{c+1} different n-gram. So, what will be the probability mass for them it will be N_{c+1} times c plus 1 divided by the total number of n-grams that are N that is a probability mass, yes, and this I am distributing to the words that occur c times. Now, how many words are there, there are N_c words?

So, what is the probability that each of them will get divided by N_c ? Yes, so this is the probability mass for that each of them will get. Now, what will be the effective count remember how we define the effective count c^* such that if I divided by n merely this will give me the same probability, so that will tell me c^* is N_{c+1} times c plus 1 divided by N_c and that is what we have written here. So, N_c in general is the n-grams that seen exactly c terms in the corpus. So, by using this definition I can find out what is effective count for any word that is occur N_c number of times in the corpus, so that is what we have seen.

(Refer Slide Time: 08:10)

Good Turing Estimation

Good Turing Smoothing

$$P_{GT}^*(\text{things with frequency } c) = \frac{c^*}{N}$$

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

What if $c = 0$

$$P_{GT}^*(\text{things with frequency } c) = \frac{N_1}{N}$$

where N denotes the total number of bigrams that actually occur in training

Pawan Goyal (IIT Kharagpur) Language Modelling: Advanced Smoothing Model Week 3: Lecture 1 5 / 18

So, this is the probability for the words things let us in c times c^* n , yes, c^* is my nothing, but my effective n-gram count. Now, what about the words that will not seen at all that are the words that that or whenever I am saying words here it means the n terms in general. So, what about the n-grams that did not occur at all matter in data? So, from my definition, I will use the probability mass of the words that was seen once. So, what

will be the probability mass? So, there are by definition there are N_1 words that was seen once. So, each one of them occurs once. So, what is the probability mass? It is N_1 by N , yes, and this probability mass I will give to the words that was seen 0 times yes.

Now, you might have a question how much of this each of them individual, we will get. So, this is the probability mass that are the word the n-grams are did not occurs in the corpus to get that we will get, but how much of this individually each of them will get, so if you can just divided by the number of n-grams that did not occur in the corpus. So, we will see some by some example how we can be estimate this value. So, here if my count is 0, so Good-Turing estimate for the things that for the frequency c when $c = 0$ is N_1 by n that we just saw, so that is a effective probability mass that I will give to the all the words or n-grams they did not occur in the training data. So, this gives me nice methods of estimating the probability for the words that did not occur in my training data.

(Refer Slide Time: 10:10)

Complications

What about words with high frequency?

- For small c , $N_c > N_{c+1}$
- For large c , too jumpy

Simple Good-Turing

Replace empirical N_k with a best-fit power law once counts get unreliable

N_1 N_L

Pravin Goyal (IIT Kharagpur) | Language Modelling: Advanced Smoothing Model | Week 3: Lecture 1 | 6/18 | 7:19 PM | 10/6/2018

Now, there are certain complications that you might have when you applying this Good-Turing estimate. So, for a small c , generally you will see that N_c is greater than N_{c+1} that is a number of words having a frequency c is greater than number of words having a frequency c plus 1. So, remember what would be empirical law that states that, so that is a (Refer Time: 10:34) law that is states that there is a universal relationship. So, this is generally good, but it might happen when we go to very, very high frequencies certain frequency are not observing that, so this is possible.

So, what do we do in general? So, when we go to higher and higher frequencies for larger k 's or larger c 's here we can replace it with some sort of best fit power law and we will do some normalizations. So, instead of doing it for each individual k , I might actually just fit, so I have N_1 , N_2 and so on. And when I go to the higher values, I simply fit some sort of power law and that is so that it is normalized and I will obtain the same probability mass that is 1. So, this is power law can be fit it. In general, so in various toolkits that you might very you might use this method, they do it for only few values of k summation few values of k , maybe k is equal to 5 or k is equal to 10. Then the formula that we were using might have to be readjusted, but the basic idea was is same what we have seen here.

(Refer Slide Time: 12:02)

Good-Turing numbers: Example

22 million words of AP Neswire

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

It looks like $c^* = c - 0.75$

Count c	Good Turing c^*
0	.0000270
1	0.446
2	1.26
3	2.24
4	3.24
5	4.22
6	5.19
7	6.21
8	7.24
9	8.25

Pawan Goyal (IIT Kharagpur) Language Modelling: Advanced Smoothing Model Week 3: Lecture 1 8 / 18

So, now suppose you apply this Good-Turing smoothing methods for a corpus like AP Neswire academic press news, there are 22 million words and so that is why you are seeing for different counts what is the Good-Turing estimate that was seen, this is for bigrams. So, what yours observing in the table? So, for the bigrams that occurred 0 number of times the Good-Turing estimate was 0.0000270; for ones, 0.446; for twice 0.126. Now, so how do I actually get this number 0.0000270? So, we can just do that quickly.

So, remember for the words that occur zero times what is the total probability that we are giving them getting the probability of N_1 divided by N . Now, what is the probability

given to the individual word? So, you multiplied by N_0 ; N_0 is the number of bigrams that occurring zero times. What would be the effective count? Effective count we will remove this N , yes. So, it will be N_1 divided by N_0 . Now, what is N_1 , so you need to find out how many bigrams occurred once yes that you can find from training data how do you find N_0 that is how many bigrams did not occurred in my data.

Now, if you remember from Shakespeare corpus, so how do we find out how many bigrams did not occur at all. So, we need to see what is my vocabulary size, what are different unigrams. Suppose in my vocabulary size of V that is my number of unigrams. So, how many bigrams are possible? V square bigrams are possible. So, how many actually occurred? Suppose I know that out of this some N_b bigrams occurred in my corpus. So, V square minus N_b gives me the bigrams that did not occur at all in my corpus. So, this will become my N_0 number of bigrams that did not occurred in my corpus. So, this value I obtain by finding out N_1 number of bigrams that occur once divided by V square minus N_b , N_b is number of bigrams in total. So, that is the unique bigrams that occur in my corpus.

Now, what is the other observation that you might have from this table? While looking at this table, so for 0 and 1, this is different, but once you start seeing from 2 onwards, what is the pattern that you seen. You see that most of the effective counts are the original count minus some value like 0.75. So, you see all of these a roughly original value minus 0.2 to 75. So, now, one might are give why do not I just use c minus 0.75 in maybe give a different value to the initial two components that is also done in practice. So, this is called the absolute discounting that is what we will see next.

(Refer Slide Time: 15:40)

Absolute Discounting Interpolation

Why don't we just subtract 0.75 (or some d)?

$$P_{\text{AbsoluteDiscounting}}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w_i)$$

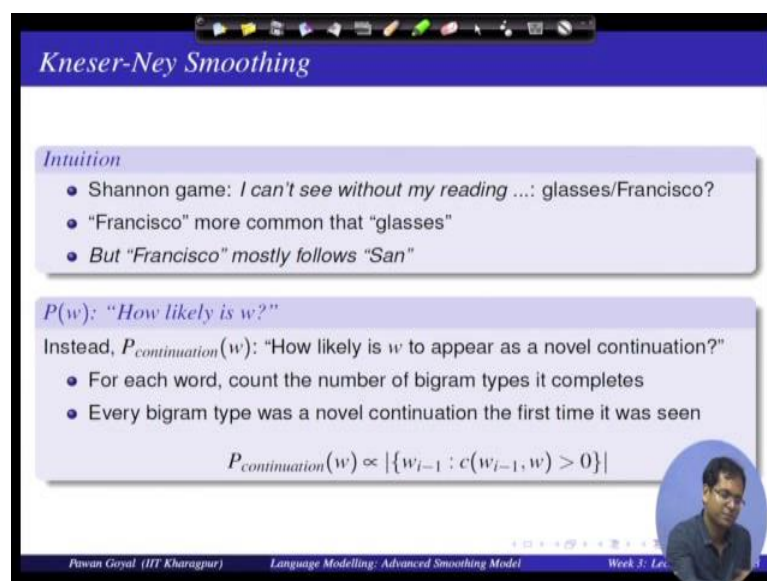
We may keep some more values of d for counts 1 and 2
But can we do better than using the regular unigram correct?

Pawan Goyal (IIT Kharagpur) Language Modelling: Advanced Smoothing Model Week 5: Lecture 1 19 / 18

So, the idea is that why do not I subtract something like 0.75 or some d from each bigram that occur say more than once or whatever. And I can have one or two different values of d for some initial bigrams, and this is called the absolute discounting methods. And you can probably interpolated with my unigram probability that we saw in the last lecture to give a better estimate, yes, this is called linear interpolation with the unigram probability of the absolute discounting method and that is the formula that we have written here. So, this is I am doing some sort of discounting from each bigram count, yes, plus I am interpolating it with my unigram probability with weight some λ w_i minus 1. Why do I need a weight here, so that when I saw this probability absolute discounting w_i given w_i minus 1 for each individual w_i this will add up to 1 that is why I need some sort of normalization constant that will make sure that this will add up to 1.

So, we can keep probably some values of d for count 1 and 2, and otherwise I can use these particular interpolation methods. Now, that is why the idea of our other advanced smoothing model comes that was our Kneser-Ney smoothing methods. So, what is that? So that is can we do better than using the regular unigram correct. So, when before we first started with a simple advanced smoothing, we were giving uniform weight to each particular n -gram uniform addition. Then from they moved on to unigram prior, we said why do we give uniformly we give a higher weight to the words that occur more often in my corpus.

(Refer Slide Time: 17:53)



Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...: glasses/Francisco?*
- "Francisco" more common than "glasses"
- But "Francisco" mostly follows "San"

$P(w)$: "How likely is w ?"

Instead, $P_{\text{continuation}}(w)$: "How likely is w to appear as a novel continuation?"

- For each word, count the number of bigram types it completes
- Every bigram type was a novel continuation the first time it was seen

$$P_{\text{continuation}}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Pawan Goyal (IIT Kharagpur) Language Modelling: Advanced Smoothing Model Week 3: Lecture 3

Now, we have seen can be do better than even this and that is why we will see the Kneser-Ney smoothing idea. So, what is the idea? So, know again let us go to the Shannon game. Suppose, at this sentence, I cannot see without my reading and have to filling this blank. And suppose possibilities are glasses and Francisco and all of us will say I should fill in glasses here. Now, take a scenario where reading glasses does not occur in my training data and reading Francisco also does not occur in my training data.

Now, if I use the previous model what will I do? The probability of filling in glasses versus Francisco will depend on their unigram probabilities. And suppose this is my data from some yours corpus, so I will find probability of Francisco is higher than probability of glasses that can very well happened and that means, I will add up filling it Francisco here there is not the correct word correct choice here. So, can we do something better than simply using their unigram probabilities that is the idea of Kneser-Ney smoothing. So, Francisco is more common than glasses in my data, but can I use this idea that in my data whenever see Francisco it occurs only after 'San', in this occurrence San Francisco.

On the other hand, whenever I see glasses, it occurs some after multiple different words. So, this is the intuition that is I will see given a word what are the other words it comes with. So, here if I take word as Francisco, it occurs maybe with only one word, yes, only one word San, but glasses occurs with many other words right, it can occur with different, different words and this. So, both glasses are occurs with ten different words

and Francisco occurs with only one different word. So, what is the intuition? If a word occurs with many other words that means, it is more likely to complete this particular bigram that we are seeing right now.

So, reading, after reading, a word is more likely to occur if it has already completed many other bigram. So, if it is more commonly used bigrams with many, many other words, this will not happen with Francisco because Francisco is used only after San. So, given a new context, probably San Francisco is not good choice, but glasses are occur many, many other different words. So, after read completely new context glasses is a better choice than Francisco.

Formally, how this method used this intuition? So, instead of using simple probability of this word unigram probability, each probability continuation word, so that is how likely is this word to appear as a novel continuation. What do I mean by novel continuation how likely it is to complete a bigram or in n-gram? So, how do I find out the probability? So, for each word to find this probability as we said we have to count the number of bigram types it completes, how many bigrams where it occurs as the last one - the final word. Now, for each bigram, whenever it occurs the first time, it was like a novel continuation, so that is what we mean by novel continuation probability here that how many bigrams it is complete. So, this will be simply, it will be proportional to the number of bigrams it is completing. So, here how many w_i minus 1 are there such that w occurs after that in my corpus. So, that is the probability of continuation.

(Refer Slide Time: 22:01)

Kneser-Ney Smoothing

How many times does w appear as a novel continuation?

$$P_{\text{continuation}}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Normalized by the total number of word bigram types

$$P_{\text{continuation}}(w) = \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|}$$

A frequent word (Francisco) occurring in only one context (San) will have a low continuation probability

Pawan Goyal (IIT Kharagpur) Language Modelling: Advanced Smoothing Model Week 3: Lecture 1 11 / 18

It is a same thing, what is the probability of continuation that should depend on the number of bigrams types it completes. Now, how should I normalize it? I want to find this probability for each word in the corpus, yes. So, this is proportional to the number of bigrams types it completes. So, I should find out in total how many bigram types are there that each word is completing, so that is effective their number of bigram types in total not totals the number of bigram types, how many different bigram types are there. So, here also we are only taking the types, how many different bigram types we are completing.

So, I will normalize it by using all possible bigrams. So, here there says all possible bigrams because I am saying the count is greater than 0. In my training data, how many bigram types I have seen. So, this will give me the probability continuation of the word the w . So, once I have found this probability, I can even go back to my unigram prior based absolute discounting method and replace $p(w_i)$ by $p_{\text{continuation}}(w_i)$ and that is my Kneser-Ney smoothing model. So, instead of using the unigram priors for smoothing, I used this Kneser-Ney smoothing by using the continuation probability. So, if I do that a frequent word like Francisco occurring only in one context of San will have a low continuation probability.

(Refer Slide Time: 23:42)

Kneser-Ney Smoothing

$$P_{KN}(w_i|w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1})P_{\text{continuation}}(w_i)$$

λ is a normalizing constant

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} |\{w : c(w_{i-1}, w) > 0\}|$$

Pawan Goyal (IIT Kharagpur) Language Modelling: Advanced Smoothing Model Week 3: Lecture 1 12 / 18

So, this will give an model with Kneser-Ney smoothing. You see so all these are there is all same this is same now instead of $p(w_i)$ we are put $p_{\text{continuation}}(w_i)$ that we get from the previous slide whatever formula we wrote in the previous slide that will give me this continuation $p(w_i)$. So, now this one question here that we discuss even in the earlier example how do we find $\lambda(w_{i-1})$. So, what was the intuition I gave? The hint was that $\lambda(w_{i-1})$ should be such that summation over k n smoothing of w_i given w_{i-1} for all w_i in my vocabulary should be 1.

So, now you take it as a simple exercise that suppose I want to find out what will be the value of $\lambda(w_{i-1})$ for this particular formula. And you can assume that d lies between 0 to 1. For this particular case, what is the value of λ ? $\lambda(w_{i-1})$ that I will get, so answer is also on this slide. So, you will get $\lambda(w_{i-1})$ of this. So, try to find out if I have to satisfy the constant of the summation of the probabilities for all the words should add up to 1, what is the value of λ that I end up with that will give you this value - particular value.

(Refer Slide Time: 25:21)

Model Combination

As N increases

- The power (expressiveness) of an N-gram model increases
- But the ability to estimate accurate parameters from sparse data decreases (i.e. the smoothing problem gets worse).

A general approach is to combine the results of multiple N-gram models.

Pawan Goyal (IIT Kharagpur) Language Modelling: Advanced Smoothing Model Week 3: Lecture 1 13 / 18

So, other than these two models of Good-Turing and Kneser-Ney smoothing, there are other smoothing models that you might be able to use. So, one simple idea instead of using a new smoothing model why do not we combine various models; so remember we always doing that somewhere. So, you computing bigram, I was trying to use the probability of the unigram, for unigram priors smoothing, now can I generalized this idea. Suppose, I am finding out the trigram probability, trigram language model; now from my data I can compute the trigram language model probabilities bigram and unigram using the MLE. Can I try to effectively combine these different probability models to find my trigrams language model?

So, there are actually two very, very popular ways in which this can be done one is called interpretation based methods, another is called back of language models. So, we will just see some simple intuition how this can be combined together. So, what we have seen as I increased my N power of my N-gram model increases we also see saw that in my when I saw the perplexity values. When I take a larger N-gram, I get lower perplexity I get better perplexity that means, the model becomes much better when I take a larger N . But what is the problem that happens if I take a larger N , remember why we did all this is smoothing because when I am taking larger N , the data becomes more and more is sparse. So, many of the actual N-grams become 0, the probability become 0.

So, at one side higher N-gram is better, a trigram model is better other side is very, very sparse. So, why cannot I take advantage of the both things that is I take a higher order N-gram model whenever I have the data, but whenever I do not have the data I try to use the low N-gram model. So, I do that togetherness single model, so that is the basic idea of these model that we will see. So, now general approach is can be combined multiple N-gram models to get a single model that can also take the advantage of larger N-gram model, but avoid this sparseness problem.

(Refer Slide Time: 27:44)

Backoff and Interpolation

It might help to use less context
when you haven't learned much about larger contexts

Backoff

- use trigram if you have good evidence
- otherwise bigram, otherwise unigram

Interpolation
mix unigram, bigram, trigram

Pawan Goyal (IIT Kharagpur) Language Modelling: Advanced Smoothing Model Week 3: Lecture 1 14 / 18

So, that is some cases might help to use less context whenever you do not have information about larger data, larger context. So, in back off what you will do suppose I am computing a trigram language model, so w_i given w_{i-1} , w_{i-2} whenever this trigram occurs in my training data I will use the probability as for my MLE. But if it does not occur in my training data I will back off to my bigram language model so that means, whenever the count of w_{i-1} , w_i is greater than 0, I will go to bi bigram; suppose that is also 0 then I will go to unigram. This is idea of back off language model. So, if you have good evidence you go to trigram, otherwise you go to bigram, otherwise you go to unigram. On the other hand, interpolation you compute all three and just interpolate then together you try to mix these together.

(Refer Slide Time: 28:46)

Backoff

Estimating $P(w_i | w_{i-2} w_{i-1})$

- If we do not have counts to compute $P(w_i | w_{i-2} w_{i-1})$ estimate this using the bigram probability $P(w_i | w_{i-1})$
- If we do not have counts to compute $P(w_i | w_{i-1})$, estimate this using the unigram probability $P(w_i)$

$P_{bo}(w_i | w_{i-2} w_{i-1}) =$

- $\hat{P}(w_i | w_{i-2} w_{i-1})$, if $c(w_{i-2} w_{i-1} w_i) > 0$
- $\lambda(w_{i-1} w_{i-2}) P_{bo}(w_i | w_{i-1})$, otherwise

where $P_{bo}(w_i | w_{i-1}) =$

- $\hat{P}(w_i | w_{i-1})$ if $c(w_{i-1} w_i) > 0$
- $\lambda(w_{i-1}) \hat{P}(w_i)$, otherwise

Handwritten notes: A diagram shows a sequence of words w_1, w_2, w_3 . A red circle highlights w_1 and w_2 with the fraction $3/5$ next to it. An arrow points from w_2 to w_3 with the fraction $2/5$ next to it. Another arrow points from w_3 to a box labeled $P(w_3 | w_i)$.

So, let us see what do will do in back off language models. So, idea is that I am computing a trigram model w_i given $w_{i-2} w_{i-1}$. So, here this is the previous word and this is previous to previous word. So, if I do not have counts for $w_{i-2} w_{i-1} w_i$, suppose this is 0 then I use probability w_i given w_{i-1} . Now, suppose count of $w_{i-1} w_i$ is also 0, then you go to probability w_i . Now, so you might add up having some problems with how do I normalize the final probability distribution that I will get. So, this is the particular formula, recursive formula of back off that you can use. So, back off for this trigram model w_i given $w_{i-2} w_{i-1}$ is $\hat{P}(w_i | w_{i-2} w_{i-1})$ if the count is greater than 0.

Now, what is \hat{P} ? This is actually similar to my maximum likelihood estimate that you will get, but it has been something has been subtracted from there. So, some probability mass has been shown from there, so there it when it can be given to the other counts. Why we need that suppose you are computing this trigram back off probability model. So, in general you are doing it after $w_{i-2} w_{i-1}$, you are finding for various words w_1, w_2, w_3 suppose this occurs three times, this occurs two times, this occurs 0 times.

So, now, once use the MLE method estimate immediately this will have a probability of 3 by 5, this I have probability of 2 by 5, suppose there is no other bigram and this will have a probability of 0. And now you want to use back off to use probability w_3

given w_{i-1} , but here itself probability mass is adding up to 1. So, how can you use the bigram probability? So, for that you might have to reduce some mass from these two values, so that is why we are writing \hat{P} and there are different ways you can reduce. So, we will take someone simple example we are doing it by some simple constant, you are reducing some simple constants, but this can be proportional also you can multiply some (Refer Time: 31:35). So, you will take that is my \hat{P} , it is a reduced probability value from my MLE estimate.

So, now, if the count is greater than 0, I take \hat{P} ; if the count is equal to 0, then I have to use the previous I have to back off to the bigram model. So, I have to use probability w_i given w_{i-1} and that is where I back off to this model, but what is interesting that you have seeing here. So, this is a recursive definition I am using the back off probability of w_i given w_{i-1} . And I multiplying with some constant again to ensure that the probability mass adds up to 1. So, the recursive definition, so how do we define probability w_i given w_{i-1} , again it is $\hat{P} w_i$ given w_{i-1} if the count is greater than 0; and $\hat{P} w_n$ if the count is equal to 0. So, I think it will help, if you take a simple example and try to see how do we actually use this definition to compute the back off probabilities.

(Refer Slide Time: 32:55)

Example Problem

In a corpus, suppose there are 4 words, a , b , c , and d . You are provided with the following counts.

n-gram	count	n-gram	count	n-gram	count
<u>aba</u>	4	<u>ba</u>	5	a	8
<u>abb</u>	0	<u>bb</u>	3	b	9
<u>abc</u>	0	<u>bc</u>	0	c	8
<u>abd</u>	0	<u>bd</u>	0	d	7

Use the recursive definition of backoff smoothing to obtain the probability distribution, $P_{\text{backoff}}(w_n | w_{n-2} w_{n-1})$, where $w_{n-1} = b$ and $w_{n-2} = a$. Also assume that $\hat{P}(x) = P(x) - 1/8$.

$P_{n-2}(w | ab)$

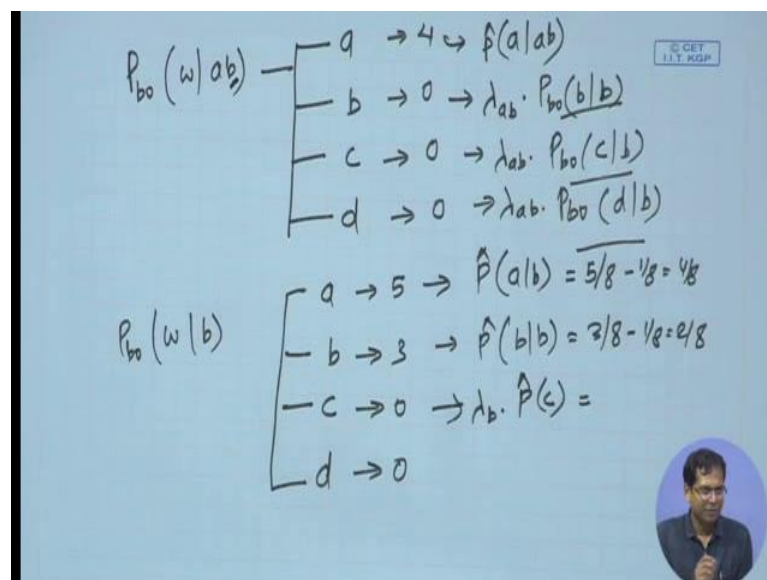
Pawan Goyal (IIT Kharagpur) Language Modelling: Advanced Smoothing Model Week 3: Lecture 1 16/18

So, let us take an example. So, here we are taking a corpus, where we are having only four words a , b , c and d and this is one data that is provided. So, what is this is data. So,

it is telling how many times a occurs after ab, it occurs four times; on the other hand, b occurs after ab - zero times, c occurs after ab - zero times. Similarly, a occurs after b - 5 times, b occur after b - 3 times, c occur after b - 0 times, and also the individual word words a occurs 8 times, b occurs 9, times c occurs 8 times d occurs 7 times.

Now, given this data, you are ask to compute the back off probability model of w_n given w_{n-1} where the previous word is b, and previous to previous word is a. And you have also told that you can use this simple definition for \hat{P} that is P_{x-1} by 8. Suppose, I want to use that to find out the back off probability distribution, so that is nothing but probability back off of word given a and b; a is pervious to previous word, and b is the previous word.

(Refer Slide Time: 34:31)



$$\begin{aligned}
 P_{bo}(w|ab) &= \begin{cases} a \rightarrow 4 \rightarrow \hat{P}(a|ab) \\ b \rightarrow 0 \rightarrow \lambda_{ab} \cdot P_{bo}(b|b) \\ c \rightarrow 0 \rightarrow \lambda_{ab} \cdot P_{bo}(c|b) \\ d \rightarrow 0 \rightarrow \lambda_{ab} \cdot P_{bo}(d|b) \end{cases} \\
 P_{bo}(w|b) &= \begin{cases} a \rightarrow 5 \rightarrow \hat{P}(a|b) = \frac{5}{8} - \frac{1}{8} = \frac{4}{8} \\ b \rightarrow 3 \rightarrow \hat{P}(b|b) = \frac{3}{8} - \frac{1}{8} = \frac{2}{8} \\ c \rightarrow 0 \rightarrow \lambda_b \cdot \hat{P}(c) = \\ d \rightarrow 0 \end{cases}
 \end{aligned}$$

So, let us try to do that. So, I want to find out the back off probability w given a, b. So, this would be w can take a, b, c and d. So, right now from my table, what are the counts, I am seeing a occurs, so this occurs four times, this occurs zero times, this occurs zero times, this occurs zero times. So, assumed by definition of back off probability, I can say that this probability will be \hat{P} a given ab, because the count is greater than the 0, but this will be λ_{ab} times previous two words ab. So, this λ_{ab} a constant times back off probability of so I will go to the previous word back off probability of b given b, this will be λ_{ab} times back off back off probability of c given the previous word b; and this will be λ_{ab} back off probability of d given b.

So, what it say is that now you have to compute the back off probability of b given b, c given b, and d given b. So, again we use the definition. So, now, I am computing back off probability of w given b, and w is equal to a, b, c and d. Now, I see a occurs 5 times, b occurs 3 times, c occurs 0 times, d occurs 0 times. So, as per my definition this will be P hat a given b what will be that that will be 5 by 8, yes minus 1 by 8 that is 4 by 8. What will be this P hat b given b 3 by 8 minus 1 by 8 that is 2 by 8? Now, what will be this? This occurs zero number of times. So, this will be lambda of b times unigram probability of c maybe P hat c. Now, what is this? We can see from my data what is P hat c. So, c occurs a times, yes and total number of, so what is the total of my unigram count, if you will add up to 32.

(Refer Slide Time: 37:19)

$$P_{b0}(w|a,b) = \begin{cases} b \rightarrow 0 \rightarrow \lambda_{ab} \cdot P_{b0}(b|b) \leftarrow \\ c \rightarrow 0 \rightarrow \lambda_{ab} \cdot P_{b0}(c|b) \leftarrow \\ d \rightarrow 0 \rightarrow \lambda_{ab} \cdot P_{b0}(d|b) \leftarrow \end{cases}$$

$$P_{b0}(w|b) = \begin{cases} a \rightarrow 5 \rightarrow \hat{P}(a|b) = \frac{5}{8} - \frac{1}{8} = \frac{4}{8} \\ b \rightarrow 3 \rightarrow \hat{P}(b|b) = \frac{3}{8} - \frac{1}{8} = \frac{2}{8} \\ c \rightarrow 0 \rightarrow \lambda_b \cdot \hat{P}(c) = \lambda_b \cdot \left[\frac{P(c) - \frac{1}{8}}{\frac{8}{32}} \right] \\ d \rightarrow 0 \rightarrow \lambda_b \cdot \hat{P}(d) = \lambda_b \cdot \left[\frac{P(d) - \frac{1}{8}}{\frac{8}{32}} \right] \end{cases}$$

$$\lambda_b \cdot \left[\frac{\frac{1}{8} + \frac{3}{32}}{\frac{8}{32}} \right] + \frac{4}{8} + \frac{2}{8} = 1$$

$$\Rightarrow \lambda_b \cdot \left[\frac{7}{24} \right] = \frac{2}{8}$$

$$\lambda_b = \frac{8}{14}$$

So, this p c the MLE varies, so it is lambda b times p c minus 1 by 8, p c is 8 by 32 yes. So, this will become 8 by 32 - 1 by 4 minus 1 by 8 that will be lambda b times 1 by 8. Similarly, for d is equal to 0, I can write lambda b times P hat d that is that is lambda b times p d minus 1 by 8 that is lambda b times p d here was 7 by 32 minus 1 by 8. So, that would be lambda b times this will be 7 minus 4 3 by 32.

So, suppose you get these values, now how will you go to the probability P w given a b you have to first find what is the back off probabilities of for these two cases c and d. So, for that you define the what is the value of lambda b, now how do you find lambda b if you normalized this right. So, you will say lambda b times 1 by 8 plus 3 by 32 plus 4 by

8 plus 2 by 8 should give me 1, right this should be 1 if you sum over all the words. So, what this is? This say lambda b times 7 by 32 is equal to 1 minus 6 by 8 that is 2 by 8. So, what is lambda b? So, we will get, you will get this 8 by 7.

So, once a we substitute add up this 8 by 7 here that will give you this probability 8. So, you will find this probability and this probability also. So, you have the back off probability for this. Now, put this back off probability here, and we have a constant how do you find this constant, again you will have to normalized it lambda a b times summation of this plus this probability should add up to one that will give you my lambda a b and also give you this probability distribution. So, that is why you find the back off probability distribution using this recursive definition.

(Refer Slide Time: 40:18)

The slide is titled "Linear Interpolation" and contains two main sections:

- Simple Interpolation**:

$$\bar{P}(w_n | w_{n-1} w_{n-2}) = \lambda_1 P(w_n | w_{n-1} w_{n-2}) + \lambda_2 P(w_n | w_{n-1}) + \lambda_3 P(w_n)$$

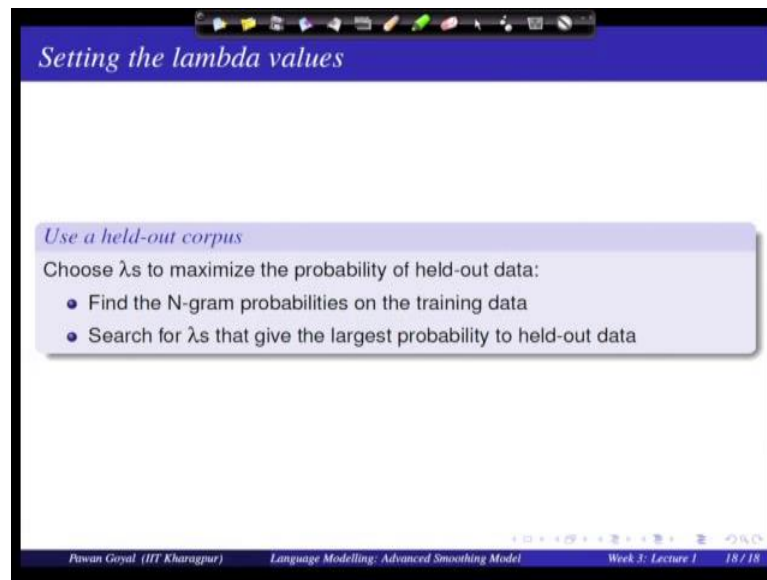
$$\sum_i \lambda_i = 1$$
- Lambdas conditional on context**:

$$\bar{P}(w_n | w_{n-1} w_{n-2}) = \lambda_1(w_{n-2}, w_{n-1}) P(w_n | w_{n-1} w_{n-2}) + \lambda_2(w_{n-2}, w_{n-1}) P(w_n | w_{n-1}) + \lambda_3(w_{n-2}, w_{n-1}) P(w_n)$$

The slide footer includes the text: "Pawan Goyal (IIT Kharagpur) Language Modelling: Advanced Smoothing Model Week 3: Lecture 1 17 / 18".

And what do you do linear interpolation, we have different unigram, bigram, trigram models, I try to interpret them in different proportions. So, what we are doing here? We have computed a trigram model, bigram model, unigram model and different proportions lambda 1, lambda 2, lambda 3 are given to this, so such that they will add up to 1. In general you can also conditional lambdas on the context, what is your previous and previous-to-previous word the lambda might depend on that. So, this is the same equation except that now lambda is a depending on the context.

(Refer Slide Time: 41:02)



Setting the lambda values

Use a held-out corpus

Choose λ s to maximize the probability of held-out data:

- Find the N-gram probabilities on the training data
- Search for λ s that give the largest probability to held-out data

Pawan Goyal (IIT Kharagpur) Language Modelling: Advanced Smoothing Model Week 3: Lecture 1 18 / 18

And now the question might be how do I choose a held-out corpus, how do I choose my lambdas, so that you will do as we say earlier you will have an held-out corpus, you will find out which values are lambda gives you the highest probability and that that lambdas you can has for your smoothing. So, that were we finish our topic of language model. So, we covered lot of simple smoothing methods, and some advanced smoothing methods and then we will now go to the next topic of so we will start with the topic of morphology, so that will be the topic of the next lecture.