

Introduction to Large Language Models (LLMs)
Prof. Tanmoy Chakraborty, Prof. Soumen Chakraborti
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi
Lecture 3
Introduction to Statistical Language Models

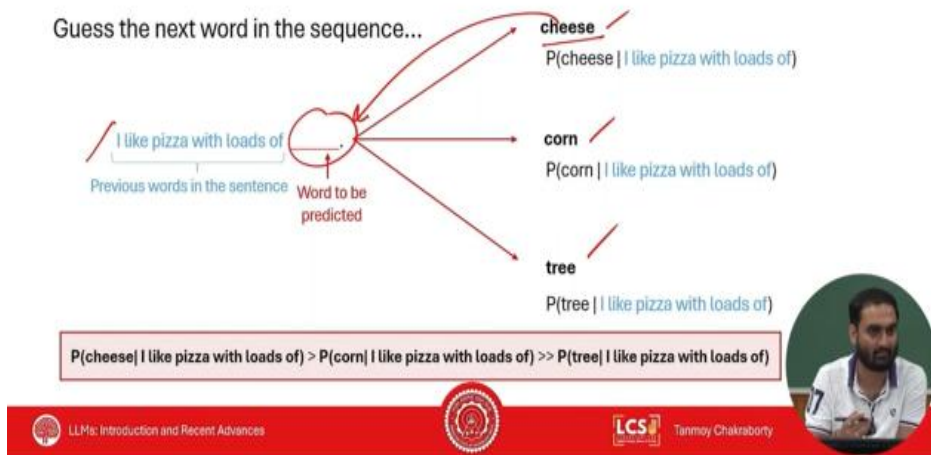
This is the third lecture of the LLM course, and today we are going to discuss language model. So the course is about large language model, let us first discuss what is language model, maybe this lecture and the next lecture, half of the next lecture we will try to finish language model, the statistical language model part of it and then we move to the concepts of word embedding and then neural language model okay.

Introduction to Statistical Language Models



Next Word Prediction

$$H(x) = -\sum p(x) \log_2 p(x) \approx 4$$



Okay now let us move to the topic statistical language model okay. So the idea of language model goes back to early 1950s okay when Claude Shannon, Shannon is the right you know Shannon's entropy right, information theory. So goes back to 1950 when Shannon introduced this game called the character guessing game. So he was essentially playing this game with his wife.

And basically the idea is that you give a sequence of characters and you ask the other person to guess what is going to be the next character. And it turned out that his wife actually guessed all the characters quite easily. And when I talk about characters, English characters, there are 26 characters plus space is another character, so 27 characters. Right? So most cases, it turned out that the wife, essentially, so she was able to guess the next character in three to four attempts.

Right? The worst case, she took eight to nine attempts. But most cases, on an average, she took three to four attempts to essentially guess the next character. OK? If you look at the number very carefully, three or four, right? Let's say four. Right? It has a very interesting relation with the entropy of English language. Right? we all know what is sinus entropy, right? What is sinus entropy? By the way what is entropy? Number of random guess measure of randomness okay average information okay anything else measure of randomness okay How do you measure it right, and if you think of this? x right as the

character right, and $p(x)$ let us assume that all these characters are equally likely there are 27 characters right.

So this $p(x)$ is 1 by 27 right. So if you do this math you see that this is essentially same as 4.3, 4.2. Why I mention this because you will see at the end of this chapter when I talk about perplexity, perplexity is a measure, It is a metric to use, perplexity is used to evaluate a language model.

We will see that there is a very nice relation between perplexity and entropy. Okay nevertheless so this is all about guessing the next character guessing the next word right and we essentially guess words almost every day in our life. Can somebody tell me where we use this guessing game or not we may be the tool or may be some system some model? So when we type something on Google search query, we see this auto completion. The Google search engine actually shows possible words that can come after the character, after the sequence of words. So this is word prediction game.

So let's look at this example. I like pizza with lots of dash. So it can be cheese, can be corn, can be tree also. I already mentioned there's a concept of vocabulary. We already have a vocabulary where there are words, possible words taken from the corpus.

So any word from the vocabulary can essentially come here. But if the language model is good enough, the language model will say that the probability that the word cheese will come here is higher than the probability of corn coming here and then the probability of tree coming here.

Probabilistic Language Models: Applications

Probabilistic language models can be used to determine the **most plausible sentence** by assigning a probability to sentences.

- **Speech Recognition**

- $P(\text{I bought fresh mangoes from the market}) \gg P(\text{I bot fresh man goes from the mar kit})$
- $P(\text{I love eating spicy samosas}) \gg P(\text{eye love eat tin spy sea some o says})$

- **Machine Translation**

- $P(\text{Heavy rainfall}) \gg P(\text{Big rainfall})$
- $P(\text{The festival of lights}) \gg P(\text{the festival of lamps})$
- $P(\text{Family gatherings}) > P(\text{Family meetings})$

- **Context Sensitive Spelling Correction**

- **Natural Language Generation**

- ...



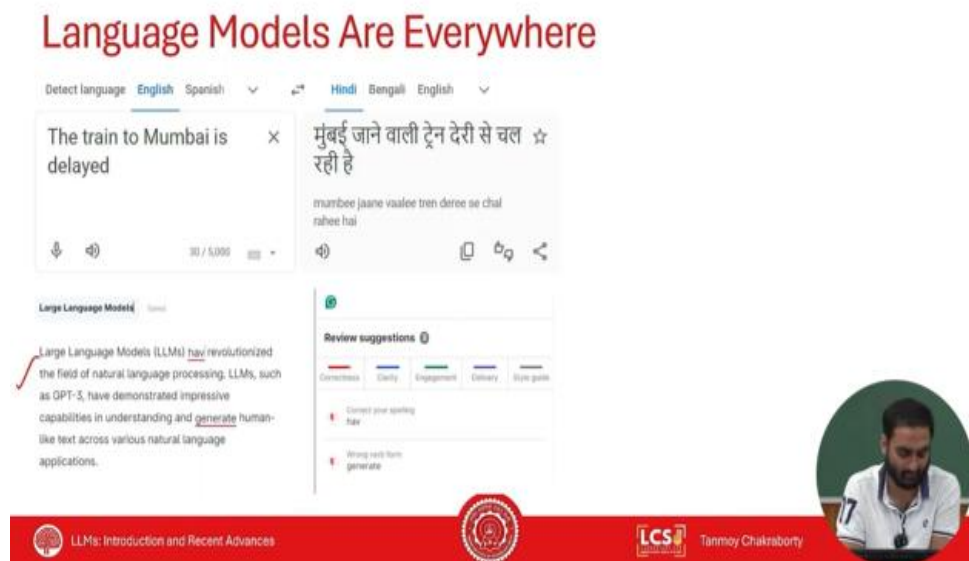
So application wise, as I mentioned, there are lots of applications. One of the applications is speech recognition. Let's say if you say that I bought fresh mangoes from the market, right? Let's say somebody spoke something and you recorded it and now you translate it, right? And people like, I mean, non-native English speakers, right? Their speaking is very different.

And if you translate the speech to a text, it may happen that you mistakenly consider market as ma or kit. Or let's say if you have this sentence, I love eating spicy samosa. You can translate it to I love eating spy sea some o says. Something like that. So if you have a language model, your language model will say that the probability of this sentence is higher than the probability of this sentence.

And that information can basically be useful for speech to text translation. Similarly, machine translation. Machine translation deals with translating from one language to another language. Let's say if you are confused between this, let's say heavy rainfall or big rainfall, right? Big brother, large brother, right? So the model should be able to say that heavy rainfall is highly likely than big rainfall or the festival of lights is highly likely than the festival of lamps, right? And so on and so forth. Remember, here, we are not only looking at syntax, but also semantics.

Lights and lamps are quite synonymous. But in the context of the festival of light is highly likely than, lamp. Similarly, family gathering is highly likely than family meetings, maybe. So therefore, oftentimes, we say that language model is so simple. But language model oftentimes captures not only the grammar, but word's knowledge.

OK. It is also used in spell checking, natural language generation, and so on and so forth. There are lots of applications.



Machine translation, as I mentioned, this is one application. Let's say you are translating English to Hindi. Here you can use machine translation language model.

Let's say you are using, I mean, let's say you are checking the spelling correction in tools like Grammarly. You see the language model can be used here. Of course, the XGBD is a language model, therefore it uses not statistical language model, but the concept of language model underline the framework.

Probabilistic Language Models

- **Goal:** Calculate the probability of a sentence or sequence consisting of n words

$$P(W) = P(w_1, w_2, w_3, \dots, w_n) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_1, w_2) \times \dots \times P(w_n|w_1, \dots, w_{n-1})$$

or

- **Related Task:** Calculate the probability of the next word conditioned on the preceding words

$$P(w_6 | w_1, w_2, w_3, w_4, w_5)$$



So let's define language model. Language model is the probability distribution over a sequence of tokens.

Very simple definition. Probability distribution over a sequence of tokens. The sequence of tokens can be sequence of characters, sequence of words, sequence of subwords, sequence of numbers, it can be anything. Right so $w_1, w_2, w_3, \dots, w_n$, there are n number of tokens right. We measure the probability of the sequence.

So this is this is same as because this is a joint probability, okay. we can essentially unfold it using chain rule and then this is going to be $P(w_1) \times P(w_2|w_1) \times P(w_3|w_1, w_2) \times \dots \times P(w_n|w_1, \dots, w_{n-1})$ right. So these are all conditional probabilities and these probabilities are conditioned on the context. So the first component is conditioned on a unigram w_1 , this is conditioned on a bigram and this is conditioned on n minus 1 gram.

Lec 03 | Introduction to Statistical Language Mod...

Watch later Share

Probability of a Sentence

Let's consider the following sentence:

The monsoon season has begun ✓

- How to compute the probability of the sentence?

$$P(W) = P(\text{"The monsoon season has begun"})$$

$$= P(\text{The, monsoon, season, has, begun})$$

8:52 / 52:47

LLP Introduction to Statistical Language Modelling

LCS CC BY-NC-SA YouTube

Lec 03 | Introduction to Statistical Language Mod...

Watch later Share

Chain Rule of Probability

- Definition of **conditional probability**:

$$P(A|B) = P(A, B) / P(B)$$

Rewriting: $P(A, B) = P(A|B) P(B)$

- More variables: $P(A, B, C, D) = P(A) \cdot P(B|A) \cdot P(C|A, B) \cdot P(D|A, B, C)$

MORE VIDEOS

8:55 / 52:47

LLP Introduction to Statistical Language Modelling

LCS CC BY-NC-SA YouTube

So this is an example, the monsoon season has begun, so you can essentially compute it using the chain rule.

This is the chain rule I already mentioned okay.

Lec 03 | Introduction to Statistical Language Mod... Watch later Share

Probability of a Sequence

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

$P(W) = P(\text{"The monsoon season has begun"})$
 $= P(\text{The, monsoon, season, has, begun})$
 $= P(\text{The}) \times P(\text{monsoon} | \text{The}) \times P(\text{season} | \text{The monsoon}) \times P(\text{has} | \text{The monsoon season}) \times P(\text{begun} | \text{The monsoon season has})$

Handwritten notes:
 An arrow points from the term $P(\text{monsoon} | \text{The})$ in the equation to the text "Cn (The monsoon)".
 Another arrow points from the term $P(\text{season} | \text{The monsoon})$ to the text "Cm (The monsoon)".

MORE VIDEOS

10:38 / 52:47

LCs CC BY-NC-SA YouTube

So now the question is, how do you, let us say, how do you measure this probability, the probability of monsoon given the or probability of season given the monsoon. What do you think? you are already given a corpus that's a significantly larger size corpus right. So how do you measure this probability? It's very simple, you first measure the count of the monsoon, this is going to be the denominator, and the numerator would be count of the monsoon season. So out of all the cases where the phrase the monsoon appears, how many cases essentially the term season follows the monsoon.

Very simple. But the problem here is that as we increase the size of the context, right? So this is our context. As we increase the length of the context, it would be very difficult to find even a single appearance of the context in the corpus. Think of this one, right? Okay. So the numerator is the monsoon season has begun.

Sorry, the denominator. So now the numerator. The numerator is the monsoon season has begun. That's the numerator. And the denominator is the monsoon season has. And this is in fact small context.

If the sentence is longer, then the context size would also be longer. Isn't it? Right, so this count is going to be 0 or whatever infinity, right?

Lec 03 | Introduction to Statistical Language Mod...

Chain Rule of Probability

- Definition of **conditional probability**:

$$P(A|B) = P(A, B) / P(B)$$
 Rewriting: $P(A, B) = P(A|B) P(B)$
- More variables: $P(A, B, C, D) = P(A) \cdot P(B|A) \cdot P(C|A, B) \cdot P(D|A, B, C)$
- The **Chain Rule** in general:

$$P(x_1, x_2, x_3, \dots, x_n) = P(x_1) P(x_2|x_1) P(x_3|x_1, x_2) \dots P(x_n|x_1, \dots, x_{n-1})$$

MORE VIDEOS

10:49 / 52:47

YouTube

So if one if in this probability, if one component is 0, then the entire probability is going to be 0, right? This is the problem, okay. So, to address this problem, let us see what we can do. Can we make some assumptions? So the assumption that we make here is the Markov assumption. Some of you have taken automata theory or some other theory where we have seen a Markov process, finite state automata and so on and so forth.

What is a Markov assumption? What is let's say a first order Markov assumption? Can somebody tell me? The first order Markov assumption says that so Markov process as we all know there are, this is a state space model, right? There are states, states are connected through transitions, and these transitions, different transitions have probabilities, right? So you start from S, this is S1, S2, a state S3, S4. This is the start state, this is the end state, for example, right? And let's say you are at this state and you are confused whether you want to move to this direction or you want to move to this direction or to this direction. Okay? So best option is that you look at all the states that you have traversed through so far, right? And then you make a decision, right? But storing the entire sequence of previous states that you have already encountered would be computationally expensive because you have to store everything, right? So here you make some assumption. We say that, okay, instead of looking at all previous states, let's look at only the previous one or the previous two. right or the previous three and so on and so forth, right.


So the first order Markov assumption says that the transition probability is only dependent on the previous state, right. Second order previous two states, last two states and so on and so forth, right. Here also we say that when we measure this probability, let us say this probability, okay, we will not look at the entire context, but we will only look at previous two words or last two words, last three words and so on and so forth, right?

Lec 03 | Introduction to Statistical Language Mod... Watch later Share

Markov Assumption

Every next state depends only the previous k states

LLMs: Introduction and Recent Advances LCS Tanmay Chakraborty




Lec 03 | Introduction to Statistical Language Mod... Watch later Share

N-gram Language Models

- Let's consider the following conditional probability:

$$P(\text{begun} \mid \text{the monsoon season has}) = \begin{matrix} P(\text{begun}) \\ P(\text{has} \mid \text{the}) \\ P(\text{has} \mid \text{the monsoon}) \end{matrix}$$
- An **N-gram model** considers only the preceding **N - 1** words.
 - ✓ Unigram: $P(\text{begun})$
 - Bigram: $P(\text{begun} \mid \text{the})$
 - Trigram: $P(\text{begun} \mid \text{the monsoon})$

LLMs: Introduction and Recent Advances LCS Tanmay Chakraborty



Lec 03 | Introduction to Statistical Language Mod...

Watch later Share

N-gram Language Models

- Let's consider the following conditional probability:

$$P(\text{begun} \mid \text{the monsoon season has})$$
- An **N-gram model** considers only the preceding **N - 1 words**.
 - Unigram: $P(\text{begun})$
 - Bigram: $P(\text{begun} \mid \text{the})$
 - Trigram: $P(\text{begun} \mid \text{the monsoon})$

MORE VIDEOS

Relation between Markov model and Language Model:
 An N-gram Language Model \equiv (N - 1) order Markov Model

17:06 / 52:47

LCS CC BY-SA YouTube

So this is Markov assumption, right? And depending on the number of words, the number of previous words that you consider to measure the probability, we will essentially call this assumption as unigram assumption, bigram assumption, trigram assumption, right? Forgram assumption and so on and so forth. If it is a unigram assumption, it means all the words are independent. So the monsoon rain has begun, the probability of the sequence is equal to probability of the times probability of monsoon, times probability of has, times probability of begun.

This is called unigram assumption. Unigram assumption is equivalent to the 0th order Markov process, right, where it is not dependent on the state at all okay. So we take this, we make this assumption, n-gram is equivalent to n minus 1 Markov assumption, right, order Markov assumption okay. So, if we take the unigram assumption, then this is essentially the same as the multiplication of all the unigram probabilities. If it's a bigram assumption, then we will only look at the previous word, right If it's a trigram assumption, then we look at last two words and so on and so forth okay. So then this is equivalent to, in case of, so if you measure these conditional probabilities, in the case of unigram, this is equivalent to p of begun, right, in the case of bigram, this is equivalent to P of begun given has, yes, if it is trigram, this is begun given season has right and so on and so forth.

So we are now restricting our context size, context length, right? If we restrict the context length, what would happen? It is highly likely that you will encounter the context in the given corpus, right? So it is less likely that you will see a zero occurrence, okay? Now, here is a question. Think about it very carefully. So Unigram assumption says that all the words are independent of each other, right? But English, in English, I mean, in all the other languages also, words are not independent at all. Right? Now, as you move from unigram to bigram to trigram to fourgram to fivegram, what actually happens is that you capture more context, more context means more grammar, better grammar.

Right. So it is obvious that as we move on, right, we would be able to capture the better grammar and the English that you generate. If you use language model to generate something, you basically generate better English. But at the same time, the probability is also going to be zero. Less likely, highly likely right. So it is a trade-off if you move to this direction, to the lower order of the Markov chain, Markov assumption then the probability is not going to be 0, but the English will be essentially meaningless.

If you move to this direction, English will be better, but the probability is going to be 0, I mean likelihood-wise. So we try to restrict also to you know somewhere between 3 to 4. So generally, 3-gram or 4-gram is the right choice for a language model, okay.

Raw bigram counts (absolute measure)

	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

Raw unigram counts (absolute measure)

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Unigram and bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Previously-zero counts are in gray.



So let us now focus on bigram because bigram is something which you can easily understand right? So let us assume that our language model that we are focusing on is a bigram language model okay? So how do we measure this probability? We need the bigram count in the numerator, right, count of X_i minus 1, X_i by count of X_i minus 1 right, so, bigram count by unigram count. Clear? Bigram count by unigram count probability of monsoon given the, right, the probability of the monsoon by the probability of the right, so, bigram count and unigram count, right? So to do this, we need a table of bigram counts and unigram counts.

Okay, so this is a bigram count table. And this table is essentially constructed from the document. We are given a huge corpus. Assume that you are given a huge corpus, you basically traverse through the document, you create the vocabulary, right? And this is a bigram count table where rows are vocabulary entries, columns are also vocabulary entries, this is a square matrix. Now, each of these entries indicates the bigram count. So H27 indicates the number of times the phrase I want appears in the document.

This is asymmetric. This is not want I, this is I want count. Now this indicates how many times the phrase to it appears in the document. Obviously, you ask why this is 5ii, right? Why this is 5, what do you think? Let's say sometimes when we, let's assume that this document is essentially a translation from speech to text, right? And if somebody fumbles, so iii, right? So this is just an error, but there can be many such errors, okay? Okay, so this is the bigram count. and this is the unigram count. Unigram count is very simple, we look at all vocabulary entries and we see how many times.

Can somebody tell me, from the bigram count, can we guess the unigram count? Yes. How? Summation of. Summation of? Okay, the next question is, what do you think from this, in this table, if you look at a single row, right, some of the entries of this row is it equal to the sum of the entries of the column? Let us assume that this is exhaustive. Let us not focus on the numbers here, is just a sample number table, right? Let us assume that you have this, the actual bigram table, right, where rows are entries, columns are entries and it is a and every entry indicates the count what do you think. So what do you think the sum of these entries would it be same as the sum of these entries? No.

Are you sure? Yes. How many of you think that this is not the same? How many of you think that this is the same? Don't hesitate. The answer is yes, okay. So the answer is yes. Sum of rows, sum of row entries for a particular row, would be the same as the sum of column entries for the particular entry, right.

Now think about it, why okay, so let me explain why. Okay, so although this is not a symmetric matrix, right, but think about it. The word i right appears with so let's say you are interested in $w_1 w_2$, this is the template of the bigram, okay, and this row entries indicate those cases where i appears as the first word which is w_1 , right, all possible cases where w_1 is basically i right. Now look at this one here. This is, all possible cases where i is essentially the second word in the bigram, right. If i is the second word in the bigram, so the first word can be anything in the document, in the vocabulary list.

Yes now right, now tell me a case, tell me, can you think of a case right, or can you think of a bigram which is not considered in rows but considering columns or vice versa. Yes very good. So, first and last word right, but remember start of a token, start of a sentence is also a token and end of a sentence is also a token in the vocabulary, right? So there will be a row which corresponds to the start of a sentence there will be a row which corresponds to the end of a sentence, that case right, so the sum would be same wonderful okay let us move on. So from the bigram table and the unigram table, now we can just, you know take the ratio of this and this right and we get the bigram probability table.

Raw bigram counts (absolute measure)

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want								
to								
eat								
chinese								
food								
lunch								
spend								

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Raw unigram counts (

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Unigram and bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Previously-zero counts are in gray.



So this is the bigram probability table, where the numerator is the bigram count and the denominator is the unigram count Okay.

Look at this table very carefully. This is a subset of the table. Now think of a million-size vocabulary, a million cross a million matrix, right? Most of the entries are zero. Most of the entries are going to be zero. In fact, in this specific corpus, this Berkeley restaurant project, 99.95% entries are zero. Okay, this is not a number that I am making it right, this is the number that was already proven 99.95% of entries are 0 in this bigram matrix, okay. This is so sparse. Now what is, let us try to understand the meaning of this, 0 entries. The 0 entries are those which neighbor appeared together in the document right.

It doesn't mean that those zero entries are invalid entries, okay.

Lec 03 | Introduction to Statistical Language Mod... Watch later Share

Limitations with MLE Estimation

Problem: N-grams only work well for word prediction if the test corpus looks like the training corpus. It is often not the case in real scenarios (data sparsity problem).

Training set: <ul style="list-style-type: none"> ... enjoyed the movie ... enjoyed the food ... enjoyed the game ... enjoyed the vacation 	Test set: <ul style="list-style-type: none"> ... enjoyed the concert ... enjoyed the festival ... enjoyed the walk
--	--

Zero probability n-grams:

- $P(\text{concert} | \text{enjoyed the}) = P(\text{festival} | \text{enjoyed the}) = P(\text{walk} | \text{enjoyed the}) = 0$
- As a result, the probability of the test set will be 0.
- Perplexity cannot be computed (Cannot divide by 0).

25:28 / 52:47

LCS CC BY-NC-SA YouTube

Let me give you an example. Let's say in your document the word enjoyed right appeared with the set of words like the movie, enjoyed the food, enjoyed the game, enjoyed the vacation, right. So if you draw this matrix, you see that, let's say trigram, trigram matrix, you see that enjoyed and these trigrams, their entries are non-zero.

But in the case set, suddenly you see enjoyed the festival. in your test set which was not there in your training document. So if you look at the bigram table, the corresponding entries enjoyed the festival, corresponding entries will be 0, right. But this should not have been 0 because this is a valid phrase, right.

Raw bigram counts (absolute measure)

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want								
to								
eat								
chinese								
food								
lunch								
spend								

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Raw unigram counts (

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Unigram and bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Previously-zero counts are in gray.





But it is not the case that all the zeros actually qualify for this. Meaning it is not the case that all the zeros can be grammatically correct or syntactically semantically correct phrase, right.

Of course, most of the zeros are meaningless, right. For example, if you look at here, right, eat, want, can never appear side by side, and two verbs can never appear side by side. Food, want again never appears. So this diagram table actually tells you it tells a lot of things, although it's a very it's looks like a statistical table right. But it also indicates many important aspects. For example if let's say, if you see that let's say Chinese food, right? Chinese food, you see the probability is quite high 0.52, right? Let's say 1.2 is 0.66. So 1.2, if you see the entry, the probability is 0.66 for 1.2, you can say that, you know, this is, so 1.2 is essentially grammatically, it is basically something, it is talking about grammar. Right whereas chinese food appears a lot of tension document, it basically indicates some sort of word's knowledge, right.

What about? Let's say to food. To food is zero can you think of, can you imagine a sentence where to food appears. I can't express my love to food, okay. What else? I went to food stall. Okay, to food, I went to food stall, right.

So the number is 0 and this 0 is called contingent 0. Why this is contingent 0? The contingent 0 means it will be hard for you to think of a sentence where to food appears. If I ask you to think of a sentence where Chinese food appears, right? You can easily think of a sentence where Chinese food appears. But if I ask you to think of a sentence where, you know, where to food, the background to food appears, you will pause and think of a sentence, then you maybe then you articulate, okay? So this is called contingency law.

Lec 03 | Introduction to Statistical Language Mod...  

Limitation of N-gram Language Models





- An insufficient model of language since they are **not effective in capturing long-range dependencies present in language.**

• Example:

The **project**, which he had been working on for months, was finally **approved** by the committee.

The above example highlights the long-distance dependency between "project" and "approved", where the context provided by earlier words affects the interpretation of later parts of the sentence.

28:35 / 52:47

Limitation of N-gram Language Models

- An insufficient model of language since they are **not effective in capturing long-range dependencies present in language.**

- Example:

The **project**, which he had been working on for months, was finally **approved** by the committee.



The above example highlights the long-distance dependency between "project" and "approved", where the context provided by earlier words affects the interpretation of later parts of the sentence.

Okay, so the other problem of the N-gram language model is that this long-range dependency. Okay, so if you look at this sentence here "The project which he had been working on for months was finally approved by the committee".

Okay, now here approved is referring to the noun project. But if you take a 2-gram language model, bigram language model, trigram language model, and if you ask the trigram language model to predict, let's say this is the context till this part, and you ask the language model, a trigram language model or a 4-gram language model, then what is going to the next word? It will fail miserably. Because the last four words can't determine the word "approved" okay. Look at the last four words "on" or "for months" comma or "months comma was finally" right Now the language model will be able to say the next word is going to be "approved", right? So if you had a language model whose context size is 10, for example, right or 12, for example, then the language model may be able to guess the next word as "approved". But you know the other problem, right, count 0.

So we will see when we will talk about RNN. Specifically you know LSTM or GRU kind of models we see this kind of long-range dependencies are addressed right by this kind of gated recurrent neural network. Any query?

Lec 03 | Introduction to Statistical Language Mod... Watch later Share

Estimate N-gram Probabilities

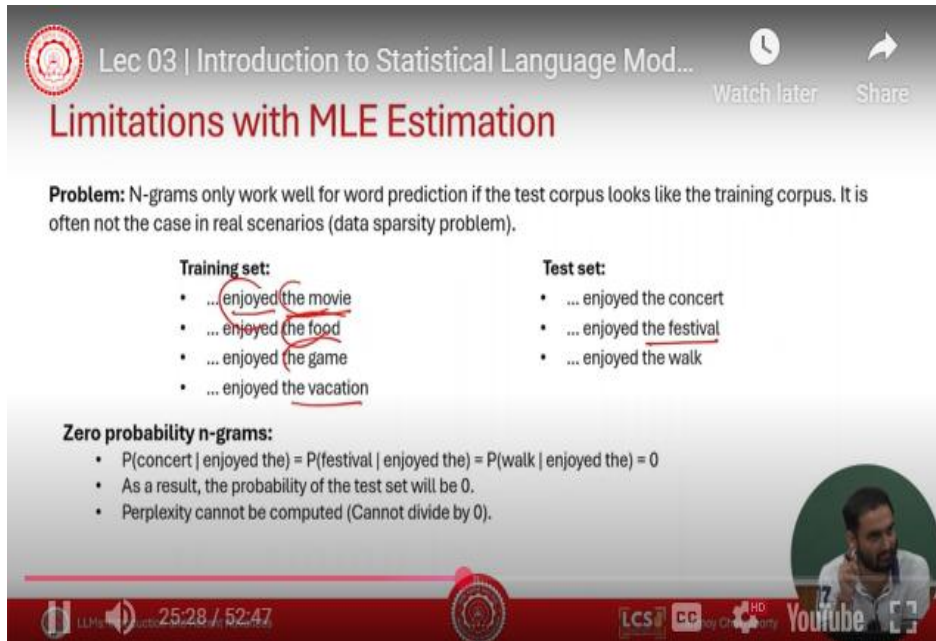
- **Maximum Likelihood Estimate (MLE):**
 - Used to estimate the parameters of a statistical model
 - Determine the most likely values of the parameters that would make the observed data most probable
- For example, **bigram probabilities** can be estimated as follows:

$$P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})} = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

30:35 / 52:47

LCS CC BY-NC-SA YouTube

Okay, so we already have seen how to estimate this probability, conditional probabilities, we use the maximum likelihood estimate MLE. And MLE says that you know if you want to predict this conditional probability, you essentially, this is the best possible probability estimate that you can think of right given a particular document okay.



The video player shows a slide titled "Limitations with MLE Estimation". The slide content is as follows:

Problem: N-grams only work well for word prediction if the test corpus looks like the training corpus. It is often not the case in real scenarios (data sparsity problem).

Training set:	Test set:
<ul style="list-style-type: none">• ... enjoyed the movie• ... enjoyed the food• ... enjoyed the game• ... enjoyed the vacation	<ul style="list-style-type: none">• ... enjoyed the concert• ... enjoyed the festival• ... enjoyed the walk

Zero probability n-grams:

- $P(\text{concert} | \text{enjoyed the}) = P(\text{festival} | \text{enjoyed the}) = P(\text{walk} | \text{enjoyed the}) = 0$
- As a result, the probability of the test set will be 0.
- Perplexity cannot be computed (Cannot divide by 0).

The video player interface includes a progress bar at 25:28 / 52:47, a volume icon, and various social media and platform logos (LCS, CC, HD, YouTube) at the bottom.

I already mentioned about the limitations of this MLE estimate in graph language model is that if you have an unknown phrase, which never appeared in the training set, right, but it's a valid phrase, you will see in the test set, it is going to produce a zero probability, right. In fact, there are cases where the test, where the words or the tokens in the test sentence right, they are not a part of the vocabulary at all.

Remember, the vocabulary is constructed from the training set, you have a training set, right, you scanned the training set, you created the vocabulary right. Now a test set is independent of the training set, it may appear, it may happen that a test set. one phrase or one word is completely unknown. Let's say LOL, right? I mentioned last day, all the social media languages, right? Then what you do? If it appears in the training set, right? And if the entries are zero in the training set, you can do some tricks and then you can adjust. But if it never appears in the training set, then in the matrix that you constructed, there is no entry.

Then what do you do? What do you think? What do you think? So this kind of case is called OOV, "out of vocabulary token". Right, so out-of-vocabulary tokens are those tokens, which are present in a test set but not in a training set. Therefore, this is not present in the bigram table or trigram table, right. So, how do you deal with this kind of words? What do you think? So I am asking those who have never taken NLP course. What do you think? How do you deal with this unknown words? Then you create a separate dictionary for the unknown words with the unknown words or what and then what? Why do you search? It never appears in the training set.

Yes. Oh okay that is a very good suggestion. Okay so what he is saying is that, Let's say the unknown word is LOL, right, which essentially is used as a replacement of whatever laugh at loud or whatever right. So it's a funny kind of phrase right. So you look at the context where the similar words appear in the training set and maybe you use that as a proxy for the probability of LOL. A good idea, but what we generally do is that now remember, for all these things to basically check the similarity and other stuff, right, you need tools, you need external theses like what net right or some other means by which you can say that this is similar to this right. So, statistically when you talk about statistical inference, so statistical inference is independent of any tool.

Do not use similarity to just predict what word can come if it is lol, don't use the lol. So what you are saying is that you look at the context and predict what can come at the place of so you look at the context and you see what are the other words which appeared after the context in the training set and you use it as a proxy of that. Like, basically, the task is to predict the word which is next to lol. So instead of this task, we create a new task to first predict what could be at the place of L1 and then using that another. Okay, that is a good suggestion, but what we do in general is that we so while creating this vocabulary right we also maintain another list which is which we call as lexicon.

What is a lexicon? A lexicon is a subset of vocabulary. Right and in a lexicon we generally store those tokens which have a frequency above a certain threshold, right? Let's say you sort all the tokens based on the frequency right, and let's say the cutoff of the frequency is five or six. So you consider all the other tokens whose frequency is below five, right? as a proxy of unknown Clear? So from your training set, now you have a lexicon and the

vocabulary minus lexicon, that subset acts as an unknown token. Right, so in your bigram count bigram table, now you instead of entries which are there in the vocabulary. Now we have entries which are in the lexicon plus an entry which is unknown, unk unk right. we use the token unk right and what is this unk? This unk essentially constitutes all this unk corresponds to all the entries of the vocabulary whose frequency is below the threshold right.

Now, whenever you encounter a new token in a test set, right, you first fetch the probability corresponding to unk from the training set and you use this to measure the probability any question yeah. Yeah, we are losing the information. Yes, yes, yes, we are losing the information. But I mean, that's the simplest option that we can think of.

So then the probability game will change. Remember the probability needs to be one and so on and so forth. Right? So that also needs to be satisfied. It's a probability. Okay. Okay, let's move on. So let's say you first scan the entire document, right? And you basically noted the frequency of all the tokens in the document, right? You sort all the tokens based on the frequency, you fix a threshold, let's say threshold is five, right? You consider all those tokens whose frequency is below five as unk. Now you have a single token called unk and the other lexicon entries are also there in the bigram table and then you do the similar kind of measurement that we did earlier. That's a very good question. So it also depends on the application right, so in general what we do, in general we consider cat and cats as two different variations and we keep those entries separately in the right now, so these are tokens, these are not types right, so the difference, these are two different vocabulary words yes.

But then, then you also need to consider the combination right. Yes. CA CA Yeah, that can also be possible. So, so what Chetan is saying is that you know, along with the vocabulary words, right, you also consider the characters and the combination of characters as tokens, right. And you put everything to the bigram table, right and let the count be zero, right. And when we do the normalization, we take care of the way we essentially take care of the zero entries that can also be possible, of course. But the problem is that maybe a possible problem would be the number of combinations that you have would be significantly large and then the size would be.

Yeah, yeah, yeah, exactly. So I mean again, as I mentioned, it depends on the application most cases we consider cat and cats as two different entries in the matrix but some cases we first do some sort of stemming and then consider. Okay let us move on. Let us now discuss some of the smoothing techniques okay. So why do we need smoothing because we have already seen the matrix is quite sparse, right, 99% entries are 0, and we need to smooth, smoothen. Somehow the simple way of smoothing is that you blindly add plus 1 to all the entries in the bigram table.

Lec 03 | Introduction to Statistical Language Mod... Watch later Share

Limitations with MLE Estimation

	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

Solution: Various smoothing techniques

40:56 / 52:47

LCS CC BY-NC-SA YouTube

Right? This is a bigram table. You add 1 to all the entries in the table. Right? If you do that, what would happen?

Lec 03 | Introduction to Statistical Language Mod... Watch later Share

Laplace Smoothing (Add-One Estimation)

- Imagine that we encountered each word (N-gram) one more time than its actual occurrence.
- Simply increase all the counts by one!
- MLE estimate (in case of bigram model)

$$P_{MLE}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

- Add-1 estimate:

$$P_{Add-1}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + |V|}$$

LLMs: Introduction and Recent Advances LCS Tanmay Chakraborty

Limitations with MLE Estimation

$$\sum_i P(i|x) = 1$$

$$\sum \frac{c(x, i) + 1}{c(x) + |V|}$$

	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

Solution: Various smoothing techniques

LLMs: Introduction and Recent Advances LCS Tanmay Chakraborty

So now, so this is called add 1 estimate. It's a new smoothing technique. We add 1 to the numerator. If we add one to the numerator right, let's say you add one here, you add one to all the bigrams, remember, bigram count is a numerator in your probability, right, you need to make sure that the conditional probability, meaning P of in this case let's say little let's look at the first row P of i given x, x can be anything here right. sum of this should be one right and so this is essentially count of x i by count of x so you added one to the numerator, so far right.

If you add one to the numerator then the probability would exit basically cross one. So we need some sort of adjustment here, right what would be the adjustment so think about it right so when you add plus one to all the entries here how many ones you are adding for every entry let's say for entry i how many ones you are adding number of columns and number of columns is the same as number of vocabulary, right. So if the vocabulary size is indicated by V, then you are adding V once, right, so if you normalize it right by V, meaning if I if you just add V to the denominator then this is going to be one, clear?

Laplace Smoothing (Add-One Estimation)

- Imagine that we encountered each word (N-gram) one more time than its actual occurrence.
- Simply increase all the counts by one!
- MLE estimate (in case of bigram model)

$$P_{MLE}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

$$\frac{C_i + k}{C_{\sum} + kV}$$

- Add-1 estimate:

$$P_{Add-1}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + |V|}$$

- Effective bigram count ($c^*(w_{i-1}, w_i)$):

$$\frac{c^*(w_{i-1}, w_i)}{c(w_{i-1})} = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + |V|}$$

Okay, so this is called add one smoothing, right? One is just a number, you can add point five, you can add point two, you can add three, four, five, right? So in general, it is called add k smoothing, right? So you add k to the numerator and kv to the denominator. Clear? Okay, so after this, once you did this, so the resultant count would change? Think about it. This is your probability, and you just made an adjustment to the probability, okay.

So this is your probability. So because of this, the bigram count will change, right? Now let's look at the resultant bigram count. So this is essentially the same as your new bigram count which is C-star and the old unigram count, because I did not change the unigram, remember this. I only change the bigram table right. I only change the bigram table, the unigram table remains the same.

So C-star is my resultant bigram count. Right, so if you do the math, so C-star is essentially going to be this, times, the unigram. So this will come here, unigram count, okay.

Comparing with Bigrams: Before and After Smoothing

	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

Add-one reconstituted counts for eight words (of $V = 1446$) in the BeRP corpus of 9332 sentences. Previously-zero counts are in gray.

Example from Speech and Language Processing book by Daniel Jurafsky and James H. Martin



LLM: Introduction and Recent Advances




Tannoy Chakraborty



So this is your new bigram count table. Just look at this table and try to remember the table briefly okay and also and what is this? This is the table after adding 1, okay, and this was the initial table.

So this was the initial bigram count table, You see a lot of zeros, right.



Lec 03 | Introduction to Statistical Language Mod...






Comparing with Bigrams: Before and After Smoothing

	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





LLM: Introduction and Recent Advances


Tannoy Chakraborty



And this is my new bigram count table, okay Let's compare these two tables side by side So earlier, all the zeros, there are lots of zeros. Now you see there is no zero here right, but there are some issues. What is the issue? The issue is that let's look at some of the high count, high frequency words, right, 827 it is now 527, right 608 it is now 238 Right? 686. This is 430. A massive drop. Right? So you know, this add one smoothing, essentially, I mean, in a funny way, we say that it essentially steals the property from reach and distributes among poor. Okay. So, the problem is that we have so many poor people in our society. Even if you steal something from reach, you'll have to steal a lot.

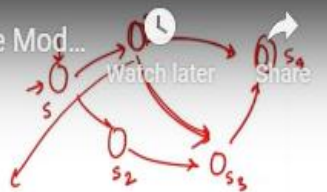
Right? So the high count will be reduced significantly. Whereas the low count look, look at this one five. Five will remain almost the same 3.8. Right? So middle income people, they are not affected. Okay. So, uh, but this is problem. This is a problem. Why is this a problem? Because now the actual count, uh, you know, uh, is not reflected properly in the resultant matrix, right? This is the problem of advanced smoothing.



Lec 03 | Introduction to Statistical Language Mod...


Estimate Conditional Probabilities








$$P(\text{begun} \mid \text{The monsoon season has}) = \frac{\text{Count}(\text{The monsoon season has begun})}{\text{Count}(\text{The monsoon season has})}$$


- **Problem:** Enough data is not available to get an accurate estimate of the above quantities.
- **Solution:** Markov Assumption



12:55 / 52:47





More General Smoothing Techniques

- Add-k smoothing:

$$P_{\text{Add-k}}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + k}{c(w_{i-1}) + k|V|}$$

$$P_{\text{Add-k}}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{|V|})}{c(w_{i-1}) + m}$$

- Unigram prior smoothing:

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

An optimal value for k or m can be determined using a held-out dataset.



So to address this, we will discuss more advanced smoothing techniques. Okay. So this is the add k smoothing that we discussed so far.

We added k to the numerator and k into v to the denominator. So if you think of k into v as m , so then k is essentially m by v . So we replace k by m times 1 by v . By the way, I am misusing the notation, actually. So in my notation, v is the size of the vocabulary and in the slide, v is the set okay, $\text{Mod } v$ is the size okay. So what am I doing now? I am replacing k by m , so, now k is going to be m by v so I now I replace k by m into 1 by v and KV by M right.

Now if you look at it carefully this 1 by V is what? 1 by V is essentially the case where this is uniform, right all the vocabulary words are equally likely right. adding uniform basically uniform some sort of uniform probability to the numerator with some constant m right. So instead of this, what you can do? You can think of, you can replace this by a probability a uniform probability. Instead of making it uniform, now you essentially replace it by a unigram probability.

Because this uniform one by V , one by V is same for all the vocabulary words. Isn't it? It is same for all the vocabulary words. But that should not have happened, right? So now what I am saying is that let us replace this by $P(w_i)$ what is $P(w_i)$, $P(w_i)$ is something that we

are predicting Now this probability is dependent on the current what that you are considering. Okay, so and then how do you measure this h and u sorry m and k . we can measure m and k using the held out data set the validation data set and you can essentially do hyper parameter tuning yes probability from the corpus right okay so uh.

Back-off and Interpolation

- As N grows larger, the N -gram model becomes more powerful. However, its capability to accurately estimate parameters decreases due to data sparsity problem.
- When we have limited knowledge about larger contexts, it can be helpful to consider less context.
- **Back-off:**
 - Opt for a trigram when there is sufficient evidence, otherwise use bigram, otherwise unigram
- **Interpolation:**
 - Mix unigram, bigram, trigram
 - Interpolation generally results in improved performance



A bit more advanced method, although they are also very simple can be back off and interpolation right. In the back off smoothing what we do? Let's say we are interested in trigram probability right and we saw the trigram probability is 0 so we basically back off and we move to bigram probability right and we see the bigram is also 0 then we move to unigram probability and so on and so forth. Again some sort of approximation. Remember this approximation may not result in a true probability value.

Remember this. This may result in some number which may not satisfy the probability rule. Right? But still in a larger context it is still useful. Interpolation is another approach.

Interpolation

Linear interpolation

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

$$\sum_i \lambda_i = 1$$

Context-dependent interpolation

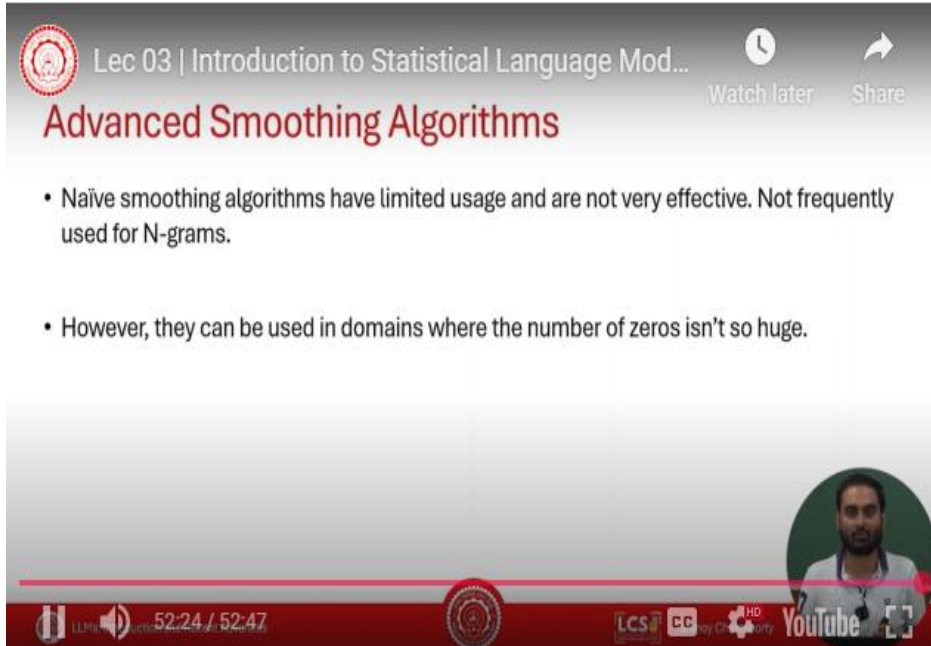
$$\hat{P}(w_n | w_{n-2} w_{n-1}) = \lambda_1(w_{n-2}^n) P(w_n | w_{n-2} w_{n-1}) + \lambda_2(w_{n-2}^n) P(w_n | w_{n-1}) + \lambda_3(w_{n-2}^n) P(w_n)$$



Interpolation says that let's say you are interested in the trigram model, right, probability of w_n given $w_{n-1} w_{n-2}$ a trigram model right. So this is, in interpolation, this is essentially a linear combination of trigram probability, bigram probability and unigram probability. Right with certain constraints λ_1 , λ_2 , λ_3 , such that sum of λ should be 1 Again, you see this is \hat{p} this is not a probability, clear.

Okay now you can again think of more advanced version where you say that instead of this raw probability count right I make this probability dependent on the context. Okay, so now this λ_1 , λ_2 , λ_3 are not normal constraints, these are functions of the context. You see λ_1 , λ_2 , λ_3 , these are functions from the context.

This is the context and this is a function. You can make it more complicated and so on and so forth. Okay, so we stop here, okay, and the next day, we will discuss two advanced smoothing techniques.



Lec 03 | Introduction to Statistical Language Mod...

Watch later Share

Advanced Smoothing Algorithms

- Naïve smoothing algorithms have limited usage and are not very effective. Not frequently used for N-grams.
- However, they can be used in domains where the number of zeros isn't so huge.

52:24 / 52:47

LCS CC BY-NC-SA YouTube

One is a good Turing method, the second one is the Naive smoothing method, right and then we will talk about perplexity, and we will see how perplexity is linked to entropy specifically cross entropy okay. Thank you.