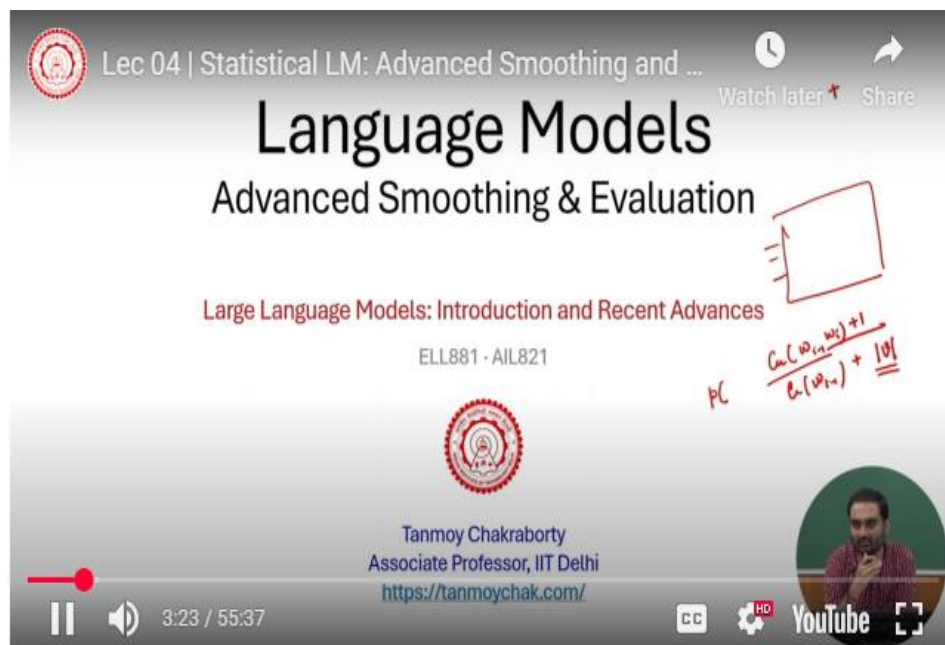


**Introduction to Large Language Models (LLMs)**  
**Prof. Tanmoy Chakraborty, Prof. Soumen Chakraborti**  
**Department of Computer Science & Engineering**  
**Indian Institute of Technology, Delhi**  
**Lecture 4**  
**Statistical LM: Advanced Smoothing and Evaluation**

Okay, so welcome again. This is the continuation of the previous lecture, we started discussing language model, right? Last class we discussed the N-gram language model, specifically the bigram language model. We focused on bigram language model and we looked at how we compute bigram language model, basically you know in any bigram language model what we do?



We compute two counts, the count of the bigram which is  $w_i$  minus 1  $w_i$  and the count of the unigram  $w_i$  minus 1. And we also discussed, we observed that in a normal bigram kind of language model, So what we do, we first compute a bigram matrix which consists of unigram Cross-unigram, kind of, rows are unigrams and columns are also unigrams, rows are essentially vocabulary words, columns are also vocabulary words and every entry in that matrix indicates the bigram count. We also observed that although the matrix is

asymmetric but the sum of rows for a particular entry and the sum of columns of a particular entry is the same. We also discussed the smoothing techniques.

We started with a simple advanced smoothing where we basically blindly added 1 to all the entries of the bigram matrix right and when we add plus 1 to all the entries right what does it mean? What does it mean? Let us say the earlier count was 2 now the count is 3, earlier count was 0 now the count is 1. What does it mean? It basically means that I have encountered that bigram one additional time, right than the count that I had earlier. Okay and we also observed that because of this, I mean when we add plus 1 to the numerator we had to add plus mod  $V$  right to the denominator and what is this mod  $V$ ? mod  $V$  is the size of the vocabulary right and this was needed because essentially this is a conditional probability and we have to make sure that the sum of this conditional probability should be 1. And we also discussed that you know, this kind of smoothing technique has some problems because what happened is that those counts, which are very high right, those counts will become very very low.

Let's say 600 will become 300 right, 500 will become 100 and so on and so forth. I also mentioned in the last class that this is essentially about stealing money from the rich and distribute it to the poor. So today we will discuss two advanced smoothing techniques, right, which will address the problems that we had earlier in advanced smoothing, right, add case smoothing, Laplace smoothing.

## 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.
- Popular Algorithms:
  - Good-Turing
  - Kneser-Ney

Use the count of things we've **seen once** to help estimate the count of things we've **never seen**



Alright, so advanced smoothing algorithms and we will look at two algorithms today, Good-Turing and Kneser-Ney, right, some people pronounce it as Neisserne, some people pronounce it as Kniezerne. So, what is the intuition here? The intuition is, we use the count of things we have seen once to estimate the count of things we have never seen.

This is the intuition.

## Notation

- $N_c$  = Frequency of frequency of  $c$
- Rohan I am I am Rohan I like to play

I	3
Rohan	2
Am	2
like	1
to	1
play	1

$N_1 = 3$   
 $N_2 = 2$   
 $N_3 = 1$

Adapted from NLP Lectures by Daniel Jurafsky



Let us look at it with an example. We define a concept of frequency of frequencies. What is a frequency of frequency? we denote this by  $N_C$ ;  $N_C$  is the frequency of frequency  $C$ , right. Let us assume that this is our corpus, right, "Rohan I am, I am Rohan I like to play", how many times the word I appear, the token I appears 3 times, Rohan 2 times, am 2 times, like 1 times and so on and so forth, right.

So what is  $N_1$ , frequency of 1, frequency of count 1 here. What is  $n_1$  here 3, what is  $n_2$  here 2, frequency of count 2, what is  $n_3$  here 1, right frequency of frequencies ok

## Good Turing Smoothing Intuition

- You are birdwatching in the Jim Corbett National Park and you have observed the following birds: 10 Flamingos, 3 Kingfishers, 2 Indian Rollers, 1 Woodpecker, 1 Peacock, 1 Crane = 18 birds
- How likely is it that the next bird you see is a woodpecker?
  - $1/18$
- How likely is it that the next bird you see is a new species -- Purple Heron or Painted Stork?
  - We will use our estimate of things we saw once to estimate the new things.
  - $3/18$  (because  $N_1 = 3$ )
- Assuming so, how likely it is that the new species is Woodpecker?
  - Must be less than  $1/18$

Adapted from NLP Lectures by Daniel Jurafsky



LLMs: Introduction and Recent Advances



Tanmoy Chakraborty

Now, what good-turing smoothing intuition? What is the intuition behind the Good-Turing smoothing technique? The intuition is that we look at the count of 1 right, count of count 1 to estimate the count of 2. Let us see. So let us assume that you know you are bird watching in, let us say, Jim Corbett National Park, and these are the birds that you have already observed, right. 10 flamingos, 2 kingfisher, 1 Indian roller, woodpecker, peacock and so on and so forth, right.

There are 18 birds that you have observed so far. So if I ask you what is the probability that the next bird that you are going to see is woodpecker. What is the probability?  $1/18$

and this is maximum likelihood estimate. What is maximum likelihood estimate? Maximum likelihood estimate says that the best probability that you can derive from the observation. And the best probability is  $1/18$  here for woodpecker right.

So maximum likelihood estimate for woodpecker is  $1/18$ . If I ask you what is the probability that the next bird that you are going to observe is purple heron say for example, a new species altogether. According to the maximum likelihood estimate this would be  $0/18$ , right? But what good-turing will say, the good-turing estimate will say that in order to estimate the count of 0, I will look at the count of 1 right. So how many count 1 is there, what is  $n_1$ , 3 right.

So to estimate this, I will use  $n_1$  right and based on that I will measure the probability. So this will be  $N_1$  by total bars 18, so  $3/18$ , okay. So, in order to measure the cases where the actual count is 0, I will look at the count of 1 and then measure the probability. So if this is  $3/18$ . If I ask you again the same question, what is the probability that the next bird is going to be woodpecker? What is the count? Would there still be  $1/18$ ? Would there still be  $1/18$ ? It is not.

Why it is not? Because we already subtracted, we already took some probability mass from  $1/18$  in order to compute this  $3/18$ . This will no longer be  $1/18$ , right. This will be less than  $1/18$ , right. What would be the value of the modified count for this, we will basically discuss okay.

## Good Turing Calculations

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

- Unseen (Purple Heron or Painted Stork)

- $C = 0$
- $\text{MLE } p = 0/18 = 0$
- $P_{GT}^*(\text{unseen}) = N_1/N = 3/18$

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

- Seen once

- $C = 1$
- $\text{MLE } p = 1/18$
- $c^*(\text{Woodpecker}) = 2 * N_2/N_1$   
 $= 2 * 1/3 = 2/3$
- $P_{GT}^*(\text{Woodpecker}) = \frac{2}{18} = 1/27$

Adapted from NLP Lectures by Daniel Jurafsky

So the good Turing probability says that things with 0 frequency the probability of those things with 0 frequency is  $N_1$  by  $N$  right.

So, in case of purple, here as I mentioned, the maximum likelihood estimate says that this is 0 but the good Turing estimate says that this is 3 by 18. For the other cases, this is going to be the modified count. The modified count  $C$ -star will be  $C$  plus 1 times  $N_C$  plus 1 by  $N_C$ . So let us say, to estimate the case where  $C$  equals 3, for example, I will look at  $C$  3 plus 1  $N_4$  by  $N_3$ . To estimate the case for  $C$  equals to 2, I look at 2 plus 1  $N_3$  by  $N_2$ .

Okay, now let's look at the probability of a woodpecker. So, according to the maximum likelihood estimate, this was earlier 1 by 18. Now this is going to be  $C$  1 plus 1,  $C$  equals to 1 here and  $N_2$  by  $N_1$  right so 2 times 1 by 3 because in  $N_2$  is  $N_2$  is 1 right look at here  $N_2$  is 1. There is only one count two, right? And what is  $N_1$ ?  $N_1$  is three. So two times one by three, two by three, two by three is the modified count.

What is the probability? Two by three by 18, because the total bar that you have observed so far is 18. So the modified probability is one by 27. The earlier probability was one by 18. Now this is one by 27. This is lower than the previous one.

right and I am not going into the theoretical details of this. If you do the theory, if you do some maths, maths is very easy. If you do the math, it will always guarantee that it will basically follow all the probability rules. Some should be 1 and so on and so forth.

Very simple idea. This is good Turing estimate.

### Good Turing Estimation

- Numbers from Church and Gale (1991)
- 22 million words of AP Newswire

$c^* = c - (n \cdot x)$

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

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

LLMs: Introduction and Recent Advances
 Tanmoy Chakraborty

So these are the numbers before good Turing estimate. Zero count, one count, two, three, four, five, six, seven, eight and these are the modified count after the Good-Turing estimate. So earlier it was zero, now this is some number, some non-negligible number .

0000270. Earlier it was one, now this is .446. The count, which was two earlier now the count is 1.26. The count, which was three earlier now this is 2.

24. Now look at the difference between these and add one smoothing the one that I discussed earlier right. After add one smoothing, we saw that the modified count is very different from the original count right. But if you look at here carefully this is not very different. After good Turing estimate the count is not very different from the previous one. But you still make sure that there is no zero.

0 is replaced by some number okay. Now if you look at this thing very carefully, the numbers very carefully, although we did some tricks but then if you look at the number very carefully, this is essentially the modified count is essentially the original count minus 0.75 roughly, roughly 0.75. The modified count is the original count minus roughly 0.75.

## Absolute Discounting Interpolation

- Adjusts the probability estimates for n-grams by discounting each count by a fixed amount (usually a small constant) before computing probabilities

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

Handwritten annotations:  $\lambda(w_{i-1})$  is written to the right of the equation. Below the equation,  $\frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})}$  is written and circled, with an arrow pointing to the denominator  $c(w_{i-1})$  in the original formula.





Lec 04 | Statistical LM: Advanced Smoothing and ...

Absolute Discounting Interpolation

- Adjusts the probability estimates for n-grams by discounting each count by a fixed amount (usually a small constant) before computing probabilities

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

Interpolation weight

unigram

- But considering the regular unigram probability has some limitations, as we will see in the upcoming slides.

MORE VIDEOS

15:31 / 55:37

YouTube

If this is the case, then why can't we just discount an absolute number, let's say 0.75 from the bigram count. Let's see what happens. Why do we need to do all these things? Ultimately, if it is only a discount of 0.75, why do we need this math? We just discount some constant  $d$  from the count and then we again modify the count.

So this is called the absolute discounting interpolation. In the absolute discounting interpolation, what we do? So  $C$  of  $w_i$  minus 1  $w_i$  by  $c$  of  $w_i$  minus 1, this is the actual probability that we are interested in. Now I am saying that you discount some factor  $d$  from the bigram count because we observe that this is essentially 0.75 discount some factor right. And since this is some sort of interpolation, if you remember interpolation back of interpolation, I mentioned last class, interpolation is what? Let us say if we are looking at bigram probability, interpolation says that you basically come up with something which is a combination of bigram and unigram right.

Here also now this corresponds to bigram and this corresponds to unigram. What is  $P$  of  $w_i$ ?  $P$  of  $w_i$  is the probability, unigram probability of  $w_i$ . What is  $w_i$ ?  $w_i$  is the second word in the bigram.  $w_i$  minus 1  $w_i$ .

Right? Unigram probability. And we can easily measure the unigram probability from the corpus. How? This unigram probability of book, the word book. We look at all the occurrences of book and you divide it by the total number of tokens. Right? And what is this lambda? Lambda is essentially again some sort of hyperparameter, whatever right that you can tune right.

Lambda can be a fixed value. Lambda can be a function of the unigram itself. In this case, in this particular formula, here, lambda is a function of the unigram itself. And how do we how do we measure this function? We have this held-out set, validation set, and we run this thing multiple times, and we decide the value of lambda. So, but this is not that easy right. This is not that easy the reason is if you look at the unigram count this  $p$  of  $w_i$  right cases like stop words right a, I, so let's say pronouns, right articles, right a, the, an right you see a lot of unigram counts right. Therefore, for stop words, frequently occurring words the unigram count will always be very high, okay.

So in the next method that we are going to discuss Knieser-Ne or Knieser-Ne there, what we will see instead of looking at or instead of measuring the unigram probability, we will measure something called a continuation probability of a unigram.

Lec 04 | Statistical LM: Advanced Smoothing and ...

## Continuation Probability

- **Intuition: Shannon game**
  - My breakfast is incomplete without a cup of ... : coffee/ Angeles?
  - Say, in the corpus "Angeles" more prevalent than "coffee"
  - However, it is important to note that "Angeles" mostly comes after "Los"
- Instead of regular unigram probability, use **continuation probability**.
  - Regular Unigram probability:  $P(w)$  : "How likely is  $w$ ?"
  - $P_{\text{continuation}}(w)$  : "How likely is  $w$  to appear as a novel continuation?"

Handwritten notes on the right side of the slide:

$w_1 w$   
 $w_2 w$   
 $w_3 w$   
 $\vdots$   
 $w_n w$

Watch later Share

MORE VIDEOS

29:08 / 55:37

YouTube

We will measure the continuation probability of a unigram. What is the continuation probability of a unigram? We all understood what is an unigram probability, right? And we also understood the problems of unigram probability. Okay? Now let's say there is a document which is related to the US, right? Where the word Los Angeles, right? Appears so many times, many, many times in the document, right? So then obviously the word Angeles will have high frequency, high unigram count. So whenever you essentially sample some number, some word from the unigram distribution, it is highly likely that the word Angeles will be sampled.

right. But in this continuation probability what we will measure? We will measure the number of unique bigrams. Please listen to this very carefully. Number of unique bigrams that this particular unigram completes. Number of unique bigrams that the given unigram which is under consideration completes.

Okay. And what is the intuition again? Again, the intuition goes back to the Shannon's guessing game, right? Let's say if I ask you this sequence, "my breakfast is incomplete without a cup of dash". If you use a unigram count, unigram probability, you will see that

the word Angeles has high probability than the word coffee. But if you sample Angelus here, if you add here, this is not going to be a correct answer. But if you look at it very carefully, the word Angelus, so this word appears only with the word loss.

Most of the times. So how many unique bigrams it completes? Only one. Whereas the word coffee appears with many other words. Hot coffee, cold coffee, regular coffee, morning coffee, breakfast coffee and so on and so forth. So instead of measuring the unigram count we will measure, something called the continuation probability of a unigram, right, which says how likely is  $w$  to appear as a novel continuation right. So we will look at all possible  $w$  xs right which is completed by the word  $w$ .

So let's say  $w_1, w, w_2, w, w_3, w, \text{dot}, \text{dot}, \text{dot}, \text{right}, w_n, w$ , so the novel continuation count of the unigram  $w$  is essentially  $n$ , okay.

## Continuation Probability

- **Intuition: Shannon game**
  - My breakfast is incomplete without a cup of ... : coffee/ Angeles?
  - Say, in the corpus "Angeles" more prevalent than "coffee"
  - However, it is important to note that "Angeles" mostly comes after "Los"
- Instead of regular unigram probability, use **continuation probability**.
  - Regular Unigram probability:  $P(w)$  : "How likely is  $w$ ?"
  - $P_{\text{continuation}}(w)$ : "How likely is  $w$  to appear as a novel continuation?"
- How to compute **continuation probability**?
  - Count how many different bigram types each word completes => Normalize by the total number of word bigram types

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



## Continuation Probability

- **Intuition: Shannon game**

- My breakfast is incomplete without a cup of ... : coffee/ Angeles?
- Say, in the corpus "Angeles" more prevalent than "coffee"
- However, it is important to note that "Angeles" mostly comes after "Los"

- Instead of regular unigram probability, use **continuation probability**.

- Regular Unigram probability:  $P(w)$  : "How likely is  $w$ ?"
- $P_{\text{continuation}}(w)$ : "How likely is  $w$  to appear as a novel continuation?"

- How to compute **continuation probability**?

- Count how many different bigram types each word completes => Normalize by the total number of word bigram types

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



LLMs: Introduction and Recent Advances



Tanmay Chakraborty

## Continuation Probability

- **Intuition: Shannon game**

- My breakfast is incomplete without a cup of ... : coffee/ Angeles?
- Say, in the corpus "Angeles" more prevalent than "coffee"
- However, it is important to note that "Angeles" mostly comes after "Los"

- Instead of regular unigram probability, use **continuation probability**.

- Regular Unigram probability:  $P(w)$  : "How likely is  $w$ ?"
- $P_{\text{continuation}}(w)$ : "How likely is  $w$  to appear as a novel continuation?"

- How to compute **continuation probability**?

- Count how many different bigram types each word completes => Normalize by the total number of word bigram types

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

A common word (Angeles) appearing in only one context (Los) is likely to have a low continuation probability.



LLMs: Introduction and Recent Advances



Tanmay Chakraborty

So this is the continuation probability of a unigram. Look at the numerator very carefully, right. We are interested in all those bigrams  $w_{i-1} w$  whose count is greater than 0 right and how many such  $w_i$ 's are there, mod of this set, so we are interested in all those  $w_{i-1}$  such that count of this is greater than 0. So this, the cardinality of this set will be the numerator, will be the count of novel continuation.

What is the denominator? What is the denominator? What do you think? What should be the denominator? Number of unique bigrams. Very good. Number of unique bigrams irrespective of the appearance of  $w$ . Right, if you look at the denominator here, count of, again, count of all the, so count of all the bigrams So this is the set of all the bigrams whose count is greater than 0 Out of all the bigrams, how many unique bigrams are completed by the unigram  $w$ , clear? If you do that, the probability of the word Angeles will be lower than the probability of the word coffee, clear?

---

## Kneser-Ney Smoothing

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

where,  $\lambda$  is a normalizing constant

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

So this is the idea of Kneser-Ney smoothing, okay. So we replace the unigram probability by the continuation probability, right and here if you look at the numerator What is this? This says max of count minus  $D$ .

So  $D$  is the absolute discount, right? Which was motivated by the previous one 0.75. So the count minus this absolute discount  $D$ , 0, because it may happen that the number is negative. If the count is 0 then 0 minus, let's say  $d$  is 0.

75 this will be negative, should not be negative. So max of these two, right, by the count of unigram. Now, can somebody tell me how to determine the value of this lambda. Let's say yeah, so in this particular formula, lambda is a function of  $w_{i-1}$ , right? Remember this is a probability, okay. I already discounted something from the first component.

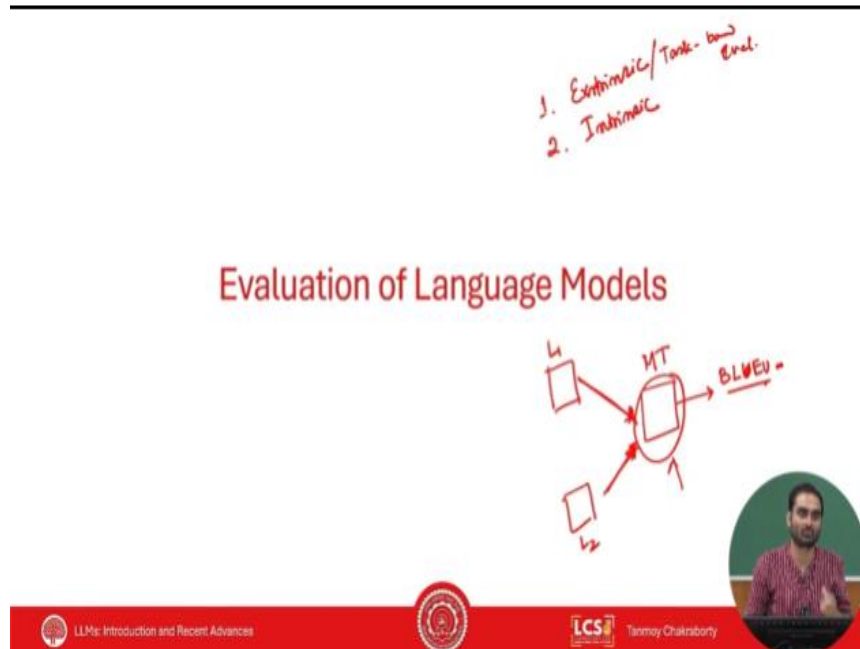
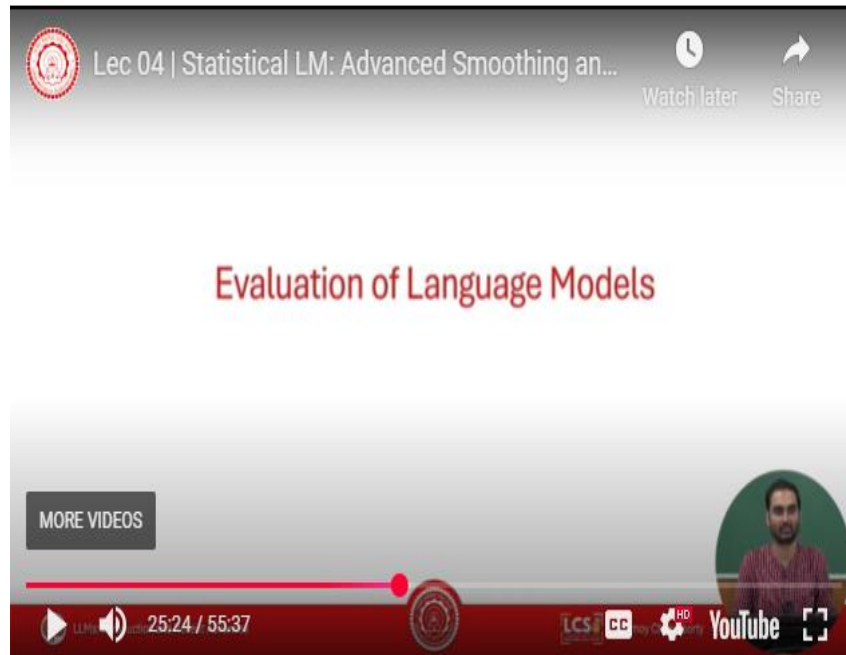
I discounted  $d$  from the first component right. So, how many discounts have happened so far, for a particular  $w_{i-1}$ . Again, think of the bigram table, okay, each row is basically  $w_{i-1}$  and column is  $w_i$ . Right, bigram table  $w_{i-1} w_i$ . Now, for every  $w_{i-1}$ , we need to make sure the sum should be one. Right now, tell me if I discount  $d$  from the bigram count, we need to adjust somewhere.

We need to adjust this so that the probability should be one. Think about it. You are almost there. So for which entries this discount is applied? Now think about the row, right? There are some entries which are non-zero.

There are some entries which are zero. So this  $D$  will only be applied to where? Non-zero and those whose value is greater than  $d$ . So, I mean this is again an approximate, I mean this should be  $d$  but again assume that this is very small number. So, for those cases whose count is greater than  $d$ , right, we will discount something from there, right, and what is going to be the denominator Numerator is  $d$ , what is the denominator, count of, what is this count of  $w_{i-1}$ ? Unigram count of previous word in terms of the table. Can you tell me in terms of the row, the count of basically the sum of values of the row. So the numerator would be  $d$  times the mod of the set for which this condition is applied by the total count because this part is discounted and this part is adjusted here.

Okay, and if you do the math, you'll see that this is, this will always guarantee that this is one. Okay, now most important part. So this was about smoothing. There are other

smoothing techniques also, but generally we follow these three smoothing techniques, advanced smoothing, ad-scale smoothing, Laplace smoothing, and Go-Turing, and Kneser-Ney name.



The most important part here is how to evaluate a language model.



Let's say you have come up with a language model which is a 5 gram language model. My language model is a 3 gram language model. How do you know that yours is better than my method? What do you think? What do you think? So the language model, when we evaluate a language model, there are two ways to evaluate a language model. One is called extrinsic evaluation.

The other is intrinsic evaluation. What is extrinsic evaluation? Extrinsic evaluation is also called task based evaluation, okay. Let's say this is a language model L1 and this is language model L2 I mentioned in the last lecture that we use this language model for downstream tasks. The task can be missing translation, task can be speech recognition, task can be spell correction, spelling correction, right? Let's say you have the task missing translation, right, for which you apply a language model. You use a language model and machine translation method model, let's say is evaluated based on some matrix. Let's say the matrix is blue, blue is a score that we generally use to evaluate the quality of a machine translation engine okay.

So what we will do? I will feed the output of L1, to machine translation model and see the value of blue score. Similarly I feed the value of output of L2 and see the score right and we will see which one is better okay. So I will feed the output of L1, I will feed the output of L2 separately in isolation and we will see for which language model, the machine translation engine will perform better, and then I will say that okay let us say L2 is better than L1 or vice versa. It is a task-based evaluation. What are the problems with task-based evaluation? What do you think? Right.

So if the quality of the machine translation model is not good, then you won't be able to essentially assess the actual performance of L1 and L2. What is the second problem? What is the second problem? The second problem is that it is not very clear which task we should choose. There are plenty of tasks, there are thousands of tasks where a language model can

be applied. Whether to choose machine translation or speech recognition system or what, it's not very clear. So what we do generally, we generally consider 10-12 different tasks and we see if a language model outperforms other language models across majority of the tasks, then we say that language model is better.

The third problem is that this extrinsic task, machine translation, speech recognition, they are time taking. They themselves are time taking because let's say they are neural models, right? So if you had to choose between L1 and L2 you may need to wait for seven days to see the performance of this downstream task and then you have to decide which one is better. Let's say you don't have time right, you have to choose very quickly among L1 and L2 right. You can't use this kind of models, this kind of frameworks right. So we will have to have some ways to intrinsically or implicitly measure the quality of a language model.

---

## Intrinsic Evaluation: Perplexity

### Intuition: The Shannon Game

- How well can we predict the next word?
  - I always order pizza with cheese and ...
  - The president of India is ...
  - I wrote a ...
- **Observation:** The more context we consider, the better the prediction.

A better text model is characterized by its ability to assign a higher probability to the correct word in a given context.



is one that best predicts an unseen test set.

The best language model is one that best predicts an unseen test set.

**Perplexity** is the inverse probability of the test data, normalized by the number of words.

- Given a sentence  $W$  consisting of  $n$  words, the perplexity is calculated as follows:

$$PP(W) = P(w_1 w_2 \dots w_n)^{-\frac{1}{n}}$$



## Perplexity and Entropy

Handwritten notes on entropy and information theory:

- Shannon's Entropy:  $H(X) = -\sum p_i \log p_i$
- Conditional Entropy:  $H(X|Y) = H(X,Y) - H(Y)$
- Joint Entropy:  $H(X,Y) = H(X) + H(Y) - H(X,Y)$
- Diagram showing a probability distribution curve (Gaussian) with parameters  $\mu$  and  $\sigma$ .
- Venn diagram illustrating the relationship between joint and conditional entropies.



And for that, what we use, we use a metric called perplexity. Again, the motivation behind perplexity comes from the Shannon's guessing game, character guessing game, word guessing game. If I ask you this sentence, I always order pizza with cheese and dash. What would be the answer? Mushroom maybe.

Cheese and mushroom. Or cheese with and what? Corn. Something like that. The set of vocabulary from where you choose the answer would be limited. Maybe 5 words or 10 words or 15 words, not more than that. But if I ask you, I wrote a dash, right, you see that the size of the vocabulary would be much, much, you know, longer. What does it mean? It means if I give you more context, if I give you more context, then the language model should be able to predict things very easily.

It is all about guessing game. Okay. So let me first define perplexity and then I will tell you the intuition behind perplexity. Okay. So what is the definition of perplexity? The definition of perplexity is very simple. So let's assume that this is your test set  $W_1, W_2$  in your test set there are  $n$  tokens.

You already have a training set. Right based on the training set you have already come up with the language model, you have already developed the language model. There is a language model  $L_1$  language model  $L_2$  right. Now you are given a test set which consists of these tokens, sequence of sentences, right, complete, incomplete sentences. what i do i measure the probability of this of the sequence of sentences based on  $L_1$  separately and based on  $L_2$  separately.

Right now based on the probability I will define the matrix for perplexity. What is perplexity? Perplexity is denoted by  $PP$  of a sequence of tokens, 1 to  $n$  is  $1/n$ th root of  $p(w_1, w_2, \dots, w_n)$ . This is same as  $p(w_1, w_2, \dots, w_n)^{1/n}$ . Agreed? Agreed? What does it mean? If a language model produces high probability, the perplexity would be low. If it produces high probability, the perplexity would be low. Now, if a language model is a bigram language model, then this would be same as? How do you compute this one? If it is bigram language model?  $(p(w_1) \times p(w_2|w_1) \times p(w_3|w_2w_1) \dots p(w_n|w_{n-1}))^{-1/n}$ .

If it is unigram then  $P$  of  $W_1$  times  $P$  of  $W_2$  dot dot dot  $P$  of  $W_n$  whole to the power minus 1 by  $n$ . And we will see that the language model which produces lower value, lower perplexity will be considered as a better model. Now why suddenly this metric? Why this one by  $n$  and you know  $n$ th root and so on right For the time being forget about this  $n$ th root forget about this one by  $n$  right one by this probability and let's assume that this is a simply  $p$  of  $w_1 w_2$  dot  $w_n$  right to the power or  $n$ th root of this probability. So what is this?  $n$ th root of this one is what? Geometric mean right.

This is geometric mean, inverse geometric mean. We know geometric mean right arithmetic mean, geometric mean right Now why suddenly this there is a very interesting connection between perplexity and entropy right and we will derive this thing here okay so We all know what is entropy? so entropy let's say there is an event  $x$ , entropy of this, this is announced entropy, entropy of this event  $x$ , let's say there are  $n$  number of outcomes and each outcome has certain probability, so this is essentially sum of  $P x_i$  or whatever  $P x_i$  log of  $P x_i$  minus okay and what is this  $x_i$ ,  $x_i$  is one outcome of  $x$  right there are  $n$  number of such outcomes okay. Now if the outcome is uniform. Let's say there are  $n$  trials and let's say there are  $c$  outcomes,  $c$  possible outcomes and each outcome appears equally equal number of times  $c$  by  $n$  or  $n$  by  $c$  versus if the outcome is biased whose entropy would be high? Entropy is a measure of randomness, whose entropy would be high? Uniform, right. We all know this famous plot right this is  $p$  and this is  $h$ , when the probability is same for all the outcomes. The entropy is maximum right all right, so we all know entropy Now we will now for a sentence, specifically for a language.

A language essentially is denoted by a sentence or a sequence of tokens. So how do we measure the entropy of a sequence of tokens? Now remember this, here there is a random variable which indicates the outcome of an event. Now here what is the random variable? A random variable indicates a sequence of words right and how do you measure this entropy here? This is same as minus of right sum of let's say  $p$  of  $w_1 w_2$  dot dot dot  $w_n$ ,

right  $\log$  of  $p$  of, right and this is for all such sequence of words which belongs to the language  $L$ , isn't it? You all agree with this equation, right? Now what we define? We define entropy rate. Why we define entropy rate? Because this is an entropy of a sequence we need to define an entropy of a word or a token right. So we define something called entropy rate or power word entropy, right. What is entropy rate or power word entropy? This is basically  $1/n$  of this  $h$  why because there are  $n$  words, this is a total entropy, power word entropy is  $1/n$  of this simple right.

So this will be  $1/n$  summation of  $p \log p$  okay This is entropy rate. Now think about it. I am only considering those cases those sentences or those sequences whose length is  $n$ . Some sequences will have length of 5, some will have length of 6, some have 10 and so on and so forth.

This is entropy rate. Now think about it. I am only considering those cases those sentences or those sequences whose length is  $n$ . Some sequences will have length of 5, some will have length of 6, some have 10 and so on and so forth. This is entropy rate. Now think about it.

I am only considering those cases those sentences or those sequences whose length is  $n$ . But in a language, in a document, there are sequences of multiple size. Some sequences will have length of 5, some will have length of 6, some have 10 and so on and so forth. But here we are only considering those sequences whose length is  $n$ . So how do we generalize it? We generalize it using this limit theory right.

we say that limit  $n$  tends to infinity right  $1/n$  of this  $p \log p$  of this okay. So remember here language is a stochastic process, right. Language is a stochastic process. What is the stochastic process? It generates tokens, so language generates tokens based on some

distribution, right and the distribution is something which we do not know.  $P$  is a distribution that we do not know, okay.

So this is entropy rate okay and we denote this entropy rate by  $H$  of  $L$ . Now  $H$  of  $L$  is this one, so far, so good. Now we will apply a theorem again I will not go into details of the theorem This is called Shannon Macmillan Bremen theorem okay The Shannon MacMillan Bremen theorem says if the stochastic process is regular, what is the regular stochastic process? It should be stationary and ergodic, again I will not go into details of this, right. If a process is a regular process right I can approximate this.

I don't need a lot of sentences right. I need a sentence whose length is significantly large. I don't need to basically take the sum over all sentences. I can approximate it by a sentence whose length is significantly large meaning it can be approximated to limit  $n$  tends to infinity  $\log$  of  $w_1 w_2 \dots w_n$  and again this  $p$  is also here. What I did? I just removed, I only removed the summation because the summation says that let us consider all the sentences right of different lengths and so on and so forth. Now using this theorem, I said no, I will not consider all the sentences, I will consider only one sentence whose length is significantly long, okay. and I will this is minus okay and I will also assume that the probability of this sentence is  $1/n$  okay.

Now these are now if you are not convinced by this, I strongly suggest that you go through this theorem and you understand it is a very standard theorem in information science, information theory, okay. So this boils down this boils down to limit  $n$  tends to infinity  $1/n \log$  of  $w_1 w_2 \dots w_n$ , happy  $p$ , yes  $p$ , sorry  $p$ , of course  $p$ , okay. Now I define another concept cross entropy, entropy we know. This is entropy. What is cross entropy? What is cross entropy? When do we need cross entropy? Let's say there are two events.

There are two random variables indicating two different events, right? And let's say you want to measure the closeness or difference or some sort of similarity between two events, right? Whether one event is independent of other events and so on and so forth. In language, nobody knows what is the actual probability or the gold probability of generating a word. Nobody knows. If you knew what is the gold probability of generating a language, then you would have solved the language generation problem.

Nobody knows how a language is generated. Right. We, we approximate this process using language model. We say that, look, we will design a language model, which will try to mimic the way language is generated. You get my point, right? So we will define something called cross entropy here, cross entropy between the actual way a language is generated and our model  $M$  okay. And what is the cross entropy formula? Sum of  $p$  of  $p_l$ ,  $p_l$  of let's say  $x_i$ ,  $\log$  of  $p_m$  of  $x_i$ .

What is  $p_l$ ?  $p_l$  is the probability obtained from the language itself. Assuming that we know the oracle. we have access to the Oracle whereas  $p_m$  is the probability obtained from the model. Let's say the model is a bigram model or trigram model and this is cross entropy right. Again if you apply this Shannon-McMillan-Griman theorem you can approximate this again I mean from here we will consider per entropy, sorry, entropy rate of per word entropy right  $\frac{1}{n}$  by  $n$  then limit and so on and so forth and apply all these things all the tricks that we applied here. So if we apply this formula if we apply the theorem again, this theorem says that the cross entropy can be written as limit  $n$  tends to infinity minus  $\frac{1}{n} \log$  of  $p_m$  I mean in our case  $x_i$  is essentially what  $w_1 w_2 \dots w_n$  okay.

So cross entropy has a very nice property by the way. Do you know the cross entropy has a bound? The original entropy  $H$  of  $P$  is the lower bound of the cross entropy  $H$  of let's say  $L$  and you can prove this very easily. Can you prove this? Can you prove this theorem? Can you prove this theorem  $H$  is bounded by  $H$ ? What is  $L$ ?  $L$  is the true the oracle



language event and what is  $M$ ?  $M$  is a model. Can you prove this? What is  $h$  of  $L$ ?  $H$  of  $L$  is  $-\sum p_i \log p_i$  right. If you replace  $p_i$  by  $p_i$  by  $m_i$  times  $m_i$  right so  $-\sum p_i \log m_i$  minus, right,  $\sum p_i \log \frac{p_i}{m_i}$ , What is this? This is cross entropy.

This is  $h(L|M)$  and what is this, by the way?  $\sum p_i \log \frac{p_i}{m_i}$ , let's see. What is this? This is KL divergence. This is KL divergence between  $P$  and  $M$  right, so  $H$  of  $L$  is essentially  $H$  of  $L$  minus KL divergence of  $L$  and  $M$ , so it means that  $H$  of  $L$  is less than equal to this. Now let's try to understand at least intuitively, try to understand this equation, right, what it says? It says that the entropy what is  $H$  of  $L$ ?  $H$  of  $L$  is the entropy of the true event  $L$  which you do not know language right. what is  $H(L, N)$  is the cross entropy between the language and the model and what is the scale divergence, scale divergence is the difference between  $L$  and  $M$ . Right so if you reduce the difference between language and the language model, the language model will come closer to the language all right.

So we have seen what is uh okay just forget about all the theories okay. Let's focus on this part, just focus on this part okay. Okay and again to simplify this, you ignore the limit, because when you consider only one sentence whose length is infinite, we can ignore the limit also. Right only this part is there  $1/n \log p$ .

Right now my claim is perplexity  $p$  is  $2$  to the power entropy. This is my claim okay. Now let's see we know the formula of public city already we have seen the formula of public city now. What is this  $h$  we have seen this is  $-\sum p_i \log p_i$  right. So you move this to the power of probability. okay, then this  $\log n$  will vanish because it's a power of two basis two.

So this is going to be  $p(w_1 w_2 \dots w_n)^{1/n}$  right. Now, pause and ponder think about it. What is entropy entropy intuitively entropy says the

average number of bits required to encode an information right log of what is log of  $1/p_i$  by  $p_i$ . What is minus log of  $p_i$ ? Number of bits equal minimum number of bits equal to encode minus  $p_i$  log of  $p_i$  is what? Expected number of bits required because  $p_i$  is essentially the probability of that outcome right. So entropy says that what is the expected number of bits required to encode an information, number of bits. What is two to the power number of bits indicating. Let's say the number of bits is three, what is two to the power three? Number of possible cases number of possible words that you can generate with three bits.

So entropy says, so the perplexity is what is the number of possible words that you can generate? Okay. Now assume that you have a sequence of tokens and you want to estimate what is going to the next word. If a language model's entropy is high, if the language model's entropy is high, so 2 to the power that will also be very high. That means there are lots of possible candidates that can come there. If a language model's entropy is low, that means your possible set of words is also low.

That means you're more uncertain about the next token. Clear? Okay, so someone can say that, why do we need perplexity? I can use entropy, I can use entropy as a measure for evaluation. But entropy and perplexity are proportional, right? If I use entropy as a measure for evaluation, it doesn't make much sense. Intuitively, if I say that, okay, the average number of bits is three for this language model, what does it mean? Versus if I say that for this language model, this language model can generate eight possible tokens for the next word versus this language model can generate three possible tokens for the next word. This makes more sense. You can say that, okay, the second language model is better than the first language model because the second language model is more certain.

Clear? So this is the beauty of perplexity. Let's look at some of the problems of statistical language model.

Lec 04 | Statistical LM: Advanced Smoot...

Watch later Share

## Problems of Statistical Language Models

- **N-gram LMs** suffer from data sparsity and limited context.
  - Predicting the next word using a fixed window of previous words.
  - Fixed Context Size: Limited to a fixed window of previous words.
- **Smoothing techniques** address data sparsity.
  - But even with smoothing, rare n-grams are hard to predict.
- Large vocabulary leads to high memory requirements.
- High computational cost for large n-grams.
- Lack of generalization to unseen word combinations.

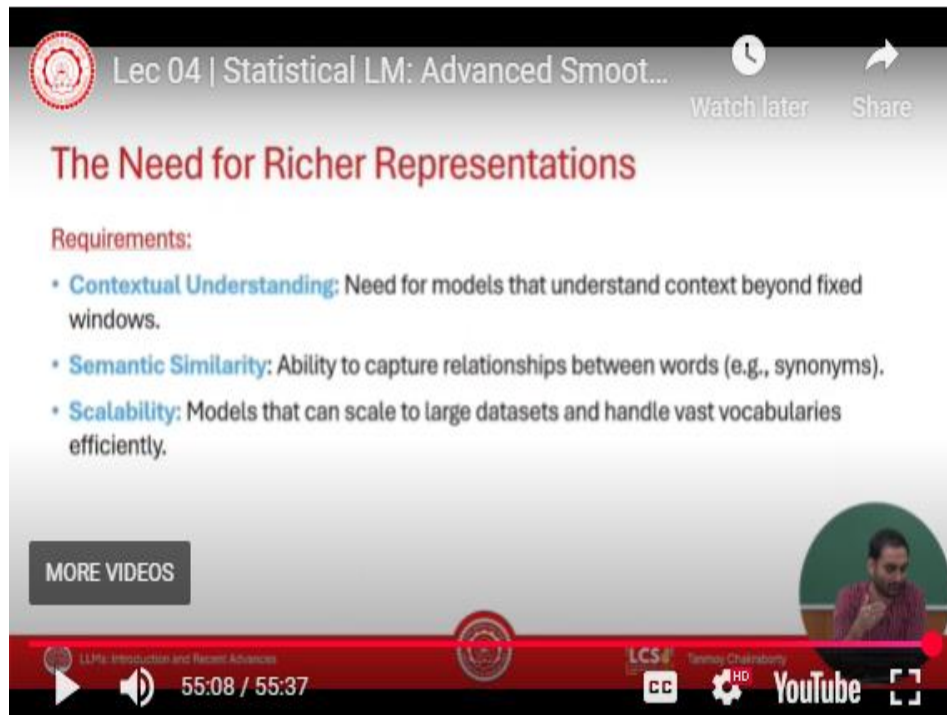
MORE VIDEOS

55:04 / 55:37

CC HD YouTube

We discussed n-gram models, we discussed smoothing techniques, we discussed evaluation measures. So n-gram models, the major problem is this fixed window size because I have to choose the window and based on that I have to measure the count and so on. What will be the size of the window is not very clear. The second problem is that You know, as the number of bigrams increases, this metric size will increase, right? Even if you do a lot of smoothing, you will still see that there are lots of small values, zero counts and so on and so forth.

It is very bad in handling out of vocabulary words, O, V. I mentioned one technique last day, but that technique is also not that great, given the amount of new words coming in every day. right and of course, it is computationally very expensive to create or to come up with this kind of engram, this bigram table, bigram or trigram table assuming that you have a trillion-size corpus, right.



Lec 04 | Statistical LM: Advanced Smoot... Watch later Share

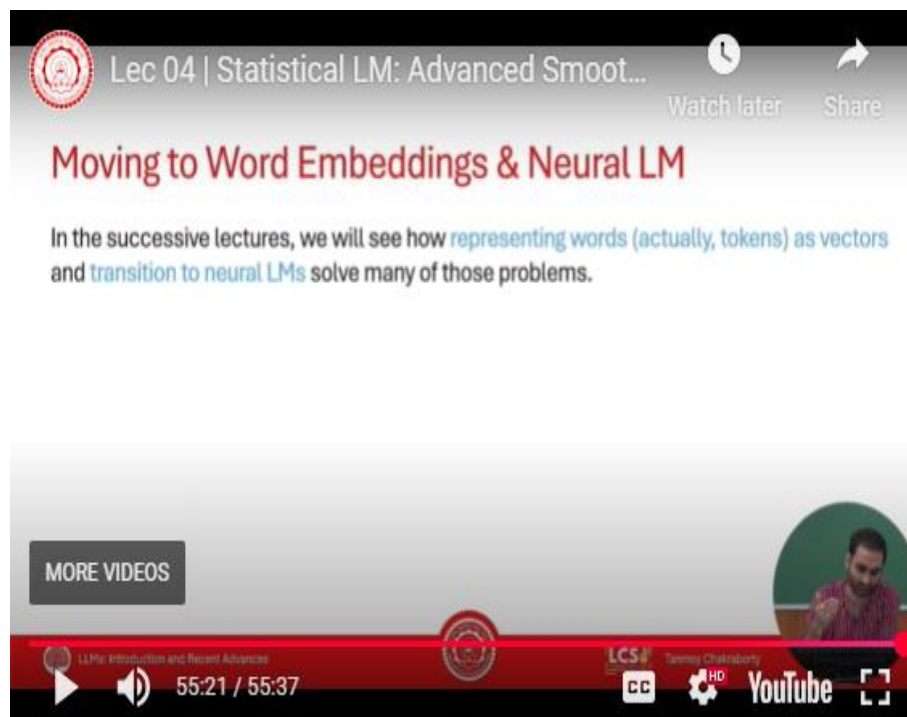
## The Need for Richer Representations

Requirements:

- **Contextual Understanding:** Need for models that understand context beyond fixed windows.
- **Semantic Similarity:** Ability to capture relationships between words (e.g., synonyms).
- **Scalability:** Models that can scale to large datasets and handle vast vocabularies efficiently.

MORE VIDEOS

LLMx Introduction and Recent Advances 55:08 / 55:37 LCS Tamas Chakraborty YouTube



Lec 04 | Statistical LM: Advanced Smoot... Watch later Share

## Moving to Word Embeddings & Neural LM

In the successive lectures, we will see how **representing words (actually, tokens) as vectors** and **transition to neural LMs** solve many of those problems.

MORE VIDEOS

LLMx Introduction and Recent Advances 55:21 / 55:37 LCS Tamas Chakraborty YouTube

So what is needed? We need some sort of better understanding of some sort of representation which understand the context properly. We will also look at how we make the entire thing scalable and so on and so forth. Thank you.