Language Models
Advanced Smoothing & Evaluation

Large Language Models: Introduction and Recent Advances

ELL881 · AIL821

Tanmoy Chakraborty
Associate Professor, IIT Delhi
https://tanmoychak.com/

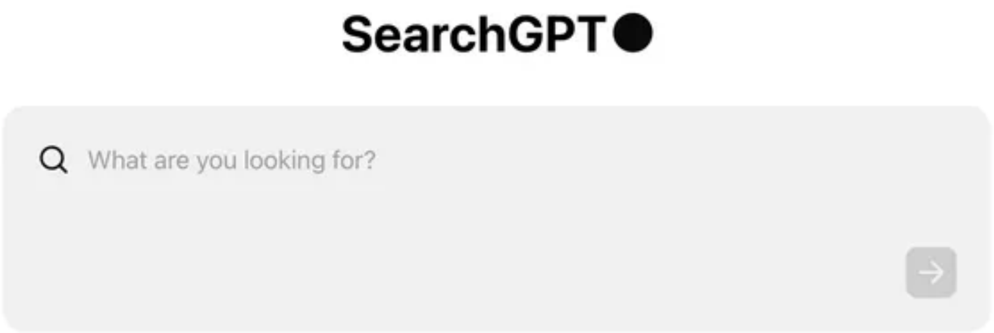
Released on
July 25, 2024

https://openai.com/index/searchgpt-prototype/

SearchGPT announced!

OpenAI announces the launch of SearchGPT, its AI-powered search engine

SearchGPT will respond to the questions with up-to-date information from the web while giving links to relevant sources.



The service is powered by the GPT-4 family of models.

A new rival to Google and Perplexity?

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.

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

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**

Notation

- N_C = **Frequency of frequency of c**

Adapted from NLP Lectures by Daniel Jurafsky

Notation

- **N_C = Frequency of frequency of c**
- Rohan I am I am Rohan I like to play

Adapted from NLP Lectures by Daniel Jurafsky

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

Adapted from NLP Lectures by Daniel Jurafsky

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

$\boxed{N_1 = 3, N_2 = 2, N_3 = 1}$

Adapted from NLP Lectures by Daniel Jurafsky

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?

Adapted from NLP Lectures by Daniel Jurafsky

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

Adapted from NLP Lectures by Daniel Jurafsky

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?

Adapted from NLP Lectures by Daniel Jurafsky

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.

Adapted from NLP Lectures by Daniel Jurafsky

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 $N_1 = 3$)

Adapted from NLP Lectures by Daniel Jurafsky

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?

Adapted from NLP Lectures by Daniel Jurafsky

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

Good Turing Calculations

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

Adapted from NLP Lectures by Daniel Jurafsky

Good Turing Calculations

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

- Unseen (Purple Heron or Painted Stork)
    - C = 0
    - MLE p = 0/18 = 0

    - P^*_{GT} (unseen) = N_1/N = 3/18

Adapted from NLP Lectures by Daniel Jurafsky

Good Turing Calculations

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

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

- Unseen (Purple Heron or Painted Stork)
  - C = 0
  - MLE p = 0/18 = 0

  - P^*_{GT} (unseen) = N_1/N = 3/18

Good Turing Calculations

- P^*_{GT}(things with zero frequency) = $\frac{N_1}{N}$
- Unseen (Purple Heron or Painted Stork)
  - C = 0
  - MLE p = 0/18 = 0
  - P^*_{GT} (unseen) = N_1/N = 3/18

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

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

Adapted from NLP Lectures by Daniel Jurafsky

Good Turing Estimation

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

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

Good Turing Estimation

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

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

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

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

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

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

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

Interpolation weight

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

unigram

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

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**.

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?"
  - $P_{continuation}(w)$: "How likely is w to appear as a novel continuation?"

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?”
  - $P_{continuation}(w)$: “How likely is w to appear as a novel continuation?”
- How to compute **continuation probability**?

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?”
  - $P_{continuation}(w)$: “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

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

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?"
  - $P_{continuation}(w)$: "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

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

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

Kneser-Ney Smoothing

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

where, λ is a normalizing constant (How to define this?)

Kneser-Ney Smoothing

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

where, λ is a normalizing constant

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

Evaluation of Language Models

Evaluation of a Language Model

- Does our language model prefer good sentences over bad ones?

Evaluation of a Language Model

- Does our language model prefer good sentences over bad ones?
  - Assign higher probability to “real” or “frequently observed” sentences than “ungrammatical” or “rarely observed” sentences

- Terminologies:
  - We optimize the parameters of our model based on data from a **training set**.
  - We assess the model's performance on unseen **test data** that is disjoint from the training data.
  - An evaluation metric provides a measure of the performance of our model on the test set.

Extrinsic Evaluation

- Measure the effectiveness of a language model by **testing their performance on different downstream NLP tasks**, such as machine translation, text classification, speech recognition.

Extrinsic Evaluation

- Measure the effectiveness of a language model by **testing their performance on different downstream NLP tasks**, such as machine translation, text classification, speech recognition.

- Let us consider two different language models: A and B
  - Select a suitable evaluation metric to assess the performance of the language models based on the chosen task.
  - Obtain the evaluation scores for A and B
  - Compare the evaluation scores for A and B

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 ...

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.

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.

Perplexity

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:

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

Perplexity

Thus, for the sentence W, perplexity is:

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

Perplexity

Thus, for the sentence W, perplexity is:

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

Applying Chain Rule:

$$PP(W) = \left(\prod \frac{1}{P(w_i \mid w_1 w_2 \ldots w_{i-1})}\right)^{\frac{1}{n}}$$

Perplexity

Thus, for the sentence W, perplexity is:

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

Applying Chain Rule:

$$PP(W) = \left(\prod \frac{1}{P(w_i \mid w_1 w_2 \ldots w_{i-1})}\right)^{\frac{1}{n}}$$

Applying Markov Assumption (n = 2), i.e. for bigram LM:

$$PP(W) = \left(\prod \frac{1}{P(w_i \mid w_{i-1})}\right)^{\frac{1}{n}}$$

Perplexity

Thus, for the sentence W, perplexity is:

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

Applying Chain Rule:

$$PP(W) = \left(\prod \frac{1}{P(w_i | w_1 w_2 \ldots w_{i-1})}\right)^{\frac{1}{n}}$$

Applying Markov Assumption (n = 2), i.e. for bigram LM:

Minimizing perplexity is the same as maximizing probability.

$$PP(W) = \left(\prod \frac{1}{P(w_i | w_{i-1})}\right)^{\frac{1}{n}}$$

Perplexity and Entropy

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.

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.

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.

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.

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.

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.

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.

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.

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.

- Move from discrete to continuous representations.
- Capture richer semantic information.
- Enable generalization to unseen data.
- Scale to large datasets.

Timeline in Language Modelling

1948

N-gram Model : Predict the next word based on the previous N-1 words

1954

Distributional Hypothesis : A word is characterized by the company it keeps

1986

Recurrent Neural Networks : Processes sequential data by using the output from previous steps as inputs for the current step

2013

Word2Vec : Distributed word representation in NLP models

2014

Attention : At each time step, the model selectively focuses on relevant words in the sequence

2017

Transformers : Uses attention and positional encoding to learn context-aware representations