Introduction to Language Models

Large Language Models: Introduction and Recent Advances

ELL881 · AIL821

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

Mistral Large 2 drops!

Mistral AI announces the release of its 123B model.

Released yesterday
July 24, 2024
https://mistral.ai/news/mistral-large-2407/

Mistral Large 2 supports **11 languages** (French, German, Spanish, Italian, Portuguese, Arabic, Hindi, Russian, Chinese, Japanese, and Korean), along with **80+ coding languages** (including Python, Java, C, C++, JavaScript, and Bash).

Its performance in **code generation**, **mathematics** and **reasoning** tasks is **comparable to larger LLMs** like GPT4o, Claude 3.5 Sonnet and Llama 3.1(405B).

Mistral Large 2 has a context window of **128k** !

Introduction to Statistical Language Models

Next Word Prediction

Guess the next word in the sequence...

I like pizza with loads of ______.

Previous words in the sentence

Word to be predicted

cheese

P(cheese | I like pizza with loads of)

corn

P(corn | I like pizza with loads of)

tree

P(tree | I like pizza with loads of)

P(cheese| I like pizza with loads of) > P(corn| I like pizza with loads of) >> P(tree| I like pizza with loads of)

Probabilistic Language Models: Applications

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

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

Probabilistic Language Models: Applications

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

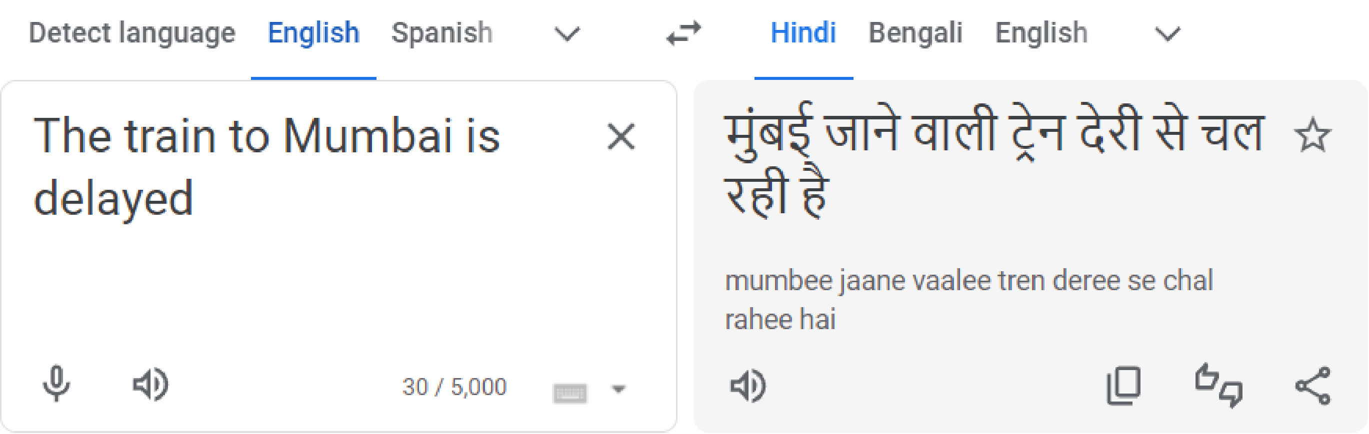
- **Speech Recognition**
  - P(I bought fresh mangoes from the market) >> P(I bot fresh man goes from the mar kit)
  - P(I love eating spicy samosas) >> P(eye love eat tin spy sea some o says)
- **Machine Translation**
  - P(Heavy rainfall) >> P(Big rainfall)
  - P(The festival of lights) >> P(the festival of lamps)
  - P(Family gatherings) > P(Family meetings)

Probabilistic Language Models: Applications

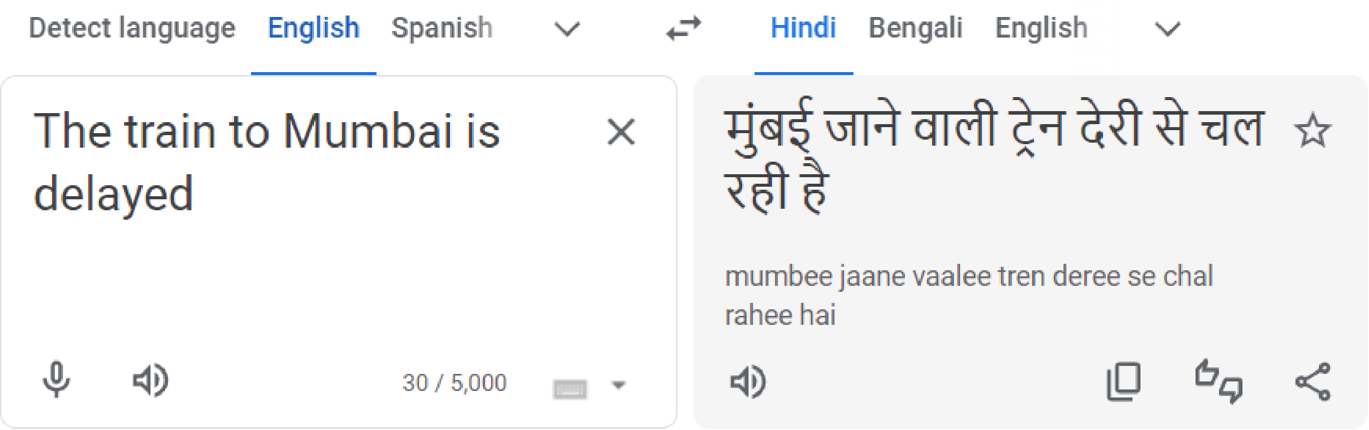
Probabilistic language models can be used to determine the **most plausible sentence** by assigning a probability to sentences.

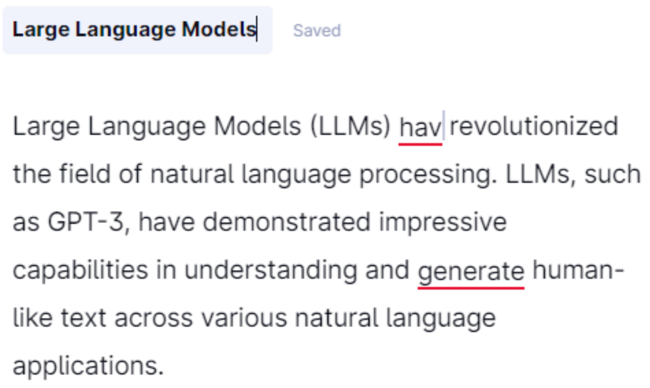
- **Speech Recognition**
  - P(I bought fresh mangoes from the market) >> P(I bot fresh man goes from the mar kit)
  - P(I love eating spicy samosas) >> P(eye love eat tin spy sea some o says)
- **Machine Translation**
  - P(Heavy rainfall) >> P(Big rainfall)
  - P(The festival of lights) >> P(the festival of lamps)
  - P(Family gatherings) > P(Family meetings)
- **Context Sensitive Spelling Correction**
- **Natural Language Generation**
- …

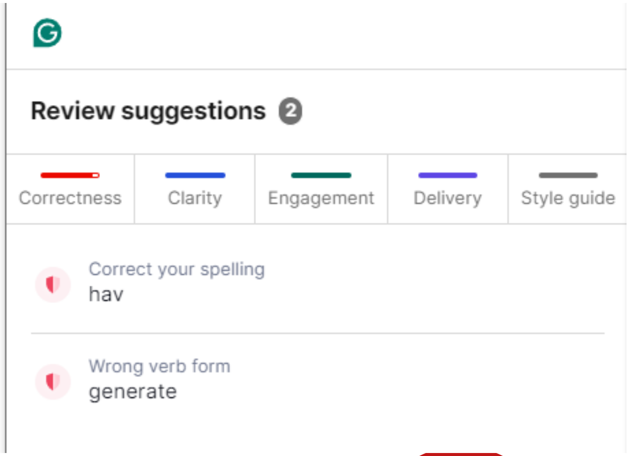
Language Models Are Everywhere



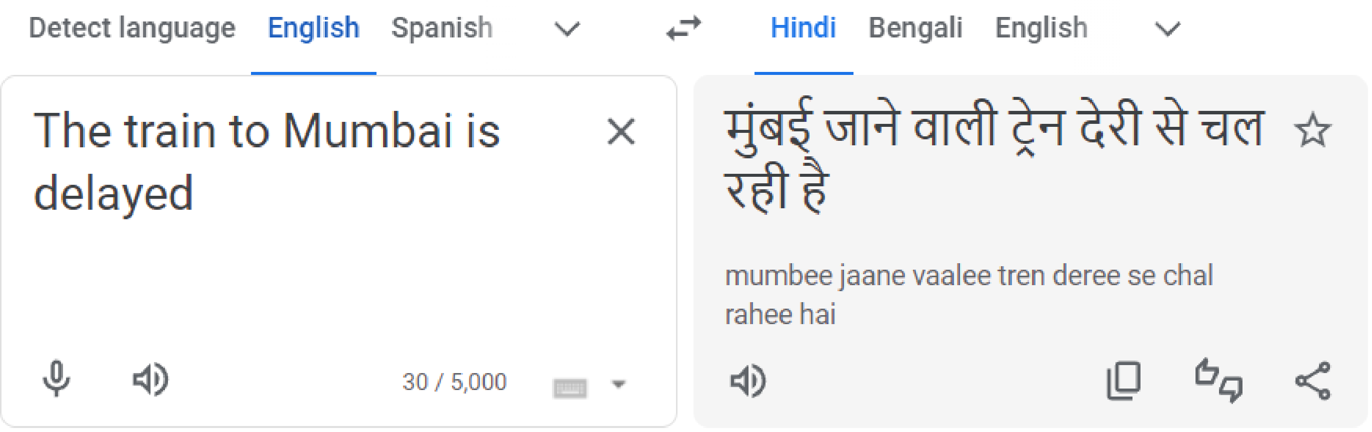
Language Models Are Everywhere

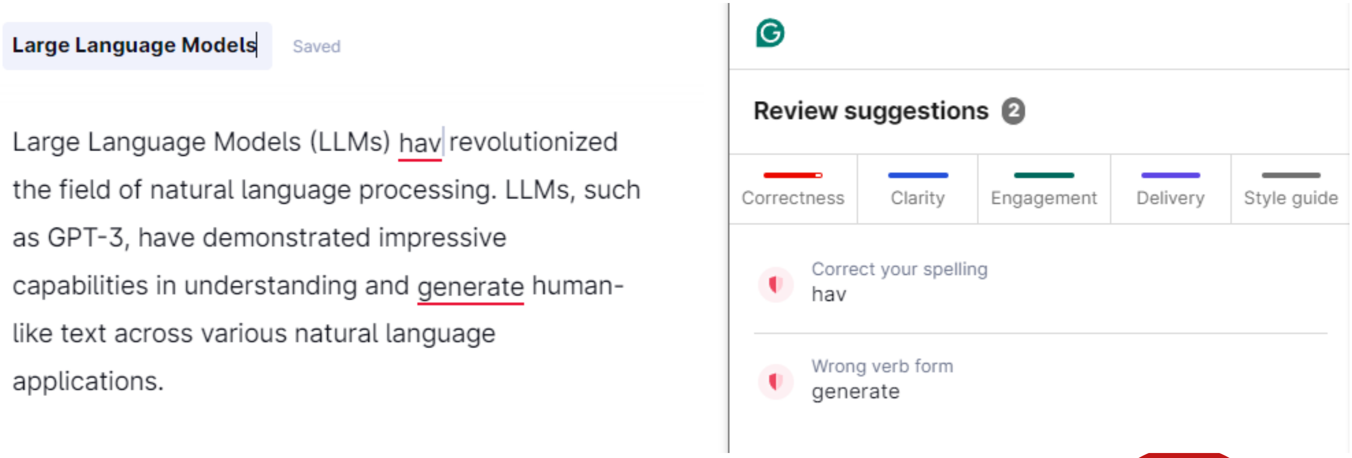


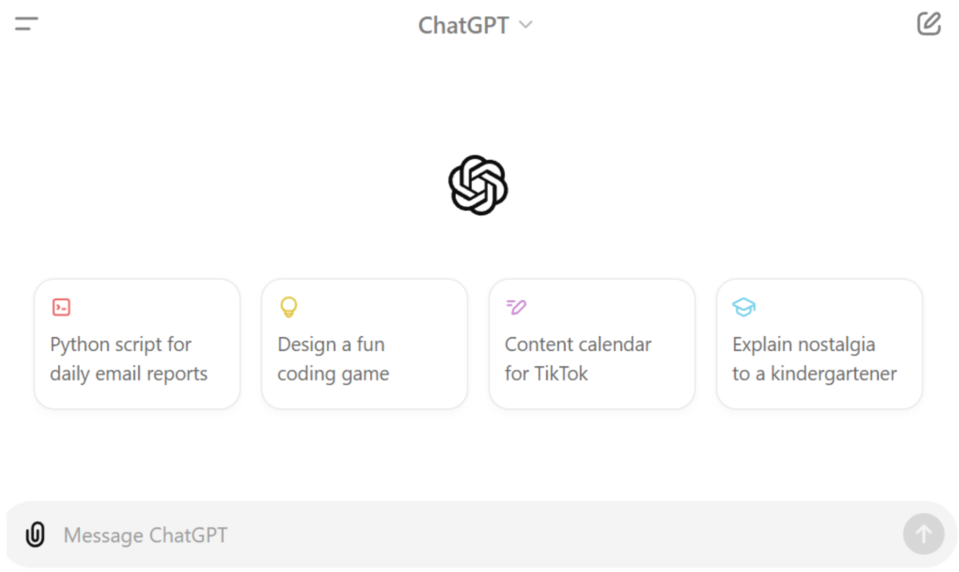




Language Models Are Everywhere







Probabilistic Language Models

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

$$P(W) = P(w_1, w_2, w_3, \ldots, w_n)$$

or

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

$$P(w_6 \mid w_1, w_2, w_3, w_4, w_5)$$

Probabilistic Language Models

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

$$P(W) = P(w_1, w_2, w_3, \ldots, w_n)$$

or

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

$$P(w_6 \mid w_1, w_2, w_3, w_4, w_5)$$

A model that calculates either of these is referred to as a **Language Model (LM).**

Probability of a Sentence

Let’s consider the following sentence:

The monsoon season has begun

- How to compute the probability of the sentence?

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

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("The monsoon season has begun")

= P(The, monsoon, season, has, begun)

We compute the above joint probability by using the principles of

Chain Rule of Probability.

Chain Rule of Probability

- Definition of **conditional probability**:

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

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

Chain Rule of Probability

- Definition of **conditional probability**:

$$P(A \mid B) = P(A, B) / P(B)$$

Rewriting: $P(A, B) = P(A \mid B)\, P(B)$

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

Chain Rule of Probability

- Definition of **conditional probability**:

$$P(A \mid B) = P(A, B) / P(B)$$

Rewriting: $P(A, B) = P(A \mid B)\, P(B)$

- More variables: $P(A,B,C,D) = P(A) \,.\, P(B \mid A) \,.\, P(C \mid A,B) \,.\, P(D \mid A,B,C)$

- The **Chain Rule** in general:

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

Probability of a Sequence

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

P(W) = P("The monsoon season has begun")

= P(The, monsoon, season, has, begun)

= P(The) x P(monsoon | The) x P(season | The monsoon) x P(has | The monsoon season) x P(begun | The monsoon season has)

Estimate Conditional Probabilities

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

Estimate Conditional Probabilities

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

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

Estimate Conditional Probabilities

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

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

Markov Assumption

Every next state depends only the previous k states

Markov Assumption

Every next state depends only the previous k states

- Chain Rule:

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

- Applying Markov Assumption we condition on only the preceding k words:

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_{i-k} \ \ldots w_{i-1})$$

Markov Assumption

Every next state depends only the previous k states

- Chain Rule:

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

- Applying Markov Assumption we condition on only the preceding k words:

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

- Probabilistic Language Models exploit the **Chain Rule of Probability** and **Markov Assumption** to build a probability distribution over sequences of words.

N-gram Language Models

- Let's consider the following conditional probability:

P(begun | the monsoon season has)

- An **N-gram model** considers only the preceding **N −1 words**.

N-gram Language Models

- Let's consider the following conditional probability:

P(begun | the monsoon season has)

- An **N-gram model** considers only the preceding **N −1 words**.
    - Unigram: P(begun)
    - Bigram: P(begun | the)
    - Trigram: P(begun | the monsoon)

N-gram Language Models

- Let's consider the following conditional probability:

P(begun | the monsoon season has)

- An **N-gram model** considers only the preceding **N −1 words**.
  - Unigram: P(begun)
  - Bigram: P(begun | the)
  - Trigram: P(begun | the monsoon)

Relation between Markov model and Language Model:

An N-gram Language Model ≡ (N −1) order Markov Model

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.

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.

Limitation of N-gram Language Models

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

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.

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

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:

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

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).

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:

- ... enjoyed the movie
- ... enjoyed the food
- ... enjoyed the game
- ... enjoyed the vacation

Test set:

- ... enjoyed the concert
- ... enjoyed the festival
- ... enjoyed the walk

Zero probability n-grams:

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

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

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!

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)

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

- Add-1 estimate:

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

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)

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

- Add-1 estimate:

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

- Effective bigram count ($c^*(w_{n-1}w_n)$):

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

Comparing with Bigrams: Before and After Smoothing

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Add-one smoothed 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.

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

Comparing with Bigrams: Before and After Smoothing

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Add-one smoothed bigram probabilities for eight of the words (out of V = 1446) in the BeRP corpus of 9332 sentences. Previously-zero probabilities are in gray.

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

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

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

More General Smoothing Techniques

- **Add-k smoothing:**

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

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

More General Smoothing Techniques

- **Add-k smoothing:**

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

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

- **Unigram prior smoothing:**

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

More General Smoothing Techniques

- **Add-k smoothing:**

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

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

- **Unigram prior smoothing:**

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

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

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.

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 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

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

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

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

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 Keoladeo 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 Keoladeo 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 Keoladeo 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 (i.e., a bird species you haven't observed yet, such as a Purple Heron or a Painted Stork)?

Adapted from NLP Lectures by Daniel Jurafsky

Good Turing Smoothing Intuition

- You are birdwatching in the Keoladeo 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 (i.e., a bird species you haven't observed yet, such as a Purple Heron or a 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 Keoladeo 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 (i.e., a bird species you haven't observed yet, such as a Purple Heron or a 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 Keoladeo 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 (i.e., a bird species you haven't observed yet, such as a Purple Heron or a 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 Keoladeo 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 (i.e., a bird species you haven't observed yet, such as a Purple Heron or a 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

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
- 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

It looks like c* = (c - 0.75)

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

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

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

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 a Language Model

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:

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

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

Minimizing perplexity is the same as maximizing probability.

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

Perplexity as Branching Factor

- Let's consider a sequence of random digits.

- What is the perplexity of this sequence according to a model that assigns a probability of p= $\frac{1}{10}$ to each digit?

Perplexity as Branching Factor

- Let's consider a sequence of random digits.

- What is the perplexity of this sequence according to a model that assigns a probability of $p = \frac{1}{10}$ to each digit?

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

$$= \left(\left(\frac{1}{10}\right)^n\right)^{-\frac{1}{n}} = 10$$

Perplexity as Branching Factor

- Let's consider a sequence of random digits.

- What is the perplexity of this sequence according to a model that assigns a probability of p= $\frac{1}{10}$ to each digit?

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

$$= ((\frac{1}{10})^n)^{-\frac{1}{n}} = 10$$

Perplexity is weighted equivalent branching factor (number of possible children).

Perplexity as Branching Factor

- Let's consider a sequence of random digits.

- What is the perplexity of this sequence according to a model that assigns a probability of $p = \frac{1}{10}$ to each digit?

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

$$= \left(\left(\frac{1}{10}\right)^{n}\right)^{-\frac{1}{n}} = 10$$

Perplexity is weighted equivalent branching factor (number of possible children).

Lower perplexity ≡ Better model

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