Language Modeling/
N-gram language model

Introduction to N-gram

Language model :

- Probabilistic models (n-gram models)
- It is the task of predicting the probability of a word or sequence of words given some context .
- Used for various NLP tasks like,
  - Text generation
  - Machine translation
  - Speech recognition
- Some of the powerful language models are GPT-3, BERT, ELMO, ROBERTa, XLNet, T5, DitilBERT, GPT-2

Probabilistic Language Modeling

- **Goal: Compute the probability of a sentence or sequence of words:**

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

Probabilistic Language Modeling

- **Goal: Compute the probability of a sentence or sequence of words:**

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

- **Related Task: probability of an upcoming word:**

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

Probabilistic Language Modeling

- **Goal: Compute the probability of a sentence or sequence of words:**

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

- **Related Task: probability of an upcoming word:**

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

A model that computes either of these is called a **language model**

Simple N-gram

- Predicting the probability of a word **w** given some history **h**, or **P(w/h)**
- Suppose the history h is:-

  Mary likes her coffee with milk and and

- Want to know the probability that the next word is **sugar :**

  P(sugar| Mary likes her coffee with milk and*)*

- How can we compute this probability?

Computing P(W)

- How to compute the joint probability?

  P(**Mary likes her coffee with milk and**)

- **Basic Idea**
  - Rely on the **Chain Rule** of Probability

Probability of an entire word sequence W

- Word sequence : $W = w_1 \ldots w_n$
- Represent the sequence of N words either as $w_1 \ldots w_n$ or w^n_1.
- Joint prob. represented as: $P(w_1, w_2, \ldots, w_n)$
- Probability of entire sequence $P(w_1, w_2, \ldots, w_n)$, use chain rule of probability :

$$
\begin{aligned}
P(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1}) \\
&= \prod_{k=1}^{n} P(w_k|w_1^{k-1})
\end{aligned}
$$

contd…

$$
\begin{aligned}
P(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1}) \\
&= \prod_{k=1}^{n} P(w_k|w_1^{k-1})
\end{aligned}
$$

- The equation above suggests that we could estimate the joint probability of an entire sequence of words by multiplying together a number of conditional probabilities.

Probability of words in sentences

$$
\begin{aligned}
P(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1}) \\
&= \prod_{k=1}^{n} P(w_k|w_1^{k-1})
\end{aligned}
$$

P(**Mary likes her coffee with milk and**) =
P(Mary) x P(likes|Mary) x P(her | Mary likes) x P(coffee | Mary likes her) x P(with | Mary likes her coffee) x P(milk | Mary likes her coffee with) x P(and | Mary likes her coffee with milk)

Estimating Probability Values

$$P(\text{sugar} \mid \text{Mary likes her coffee with milk and}) = \frac{\text{Count(Mary likes her coffee with milk and sugar)}}{\text{Count(Mary likes her coffee with milk and)}}$$

Estimating Probability Values

$$P(\text{sugar} \mid \text{Mary likes her coffee with milk and}) = \frac{\text{Count(Mary likes her coffee with milk and sugar)}}{\text{Count(Mary likes her coffee with milk and)}}$$

- What is the problem
  - We may never see enough data for estimating these

Basic Intuition

- The **intuition of the *N-gram model*** *is that instead of computing the probability of* a word given its entire history, we will **approximate the history by just the last few** words.

Markov assumption

- **Use Markov assumption-** The probability of a word depends only on the previous word

Bigram Model looks one word into the past:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k | w_{k-1})$$

Trigram model looks two words into the past:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k | w_{k-1}, w_{k-2})$$

N-gram model looks *N –1 words into the past*:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k | w_{k-1}, w_{k-2}, w_{k-3}, \ldots$$

Probability estimation for N-gram

- *Using* **Maximum Likelihood Estimation, or MLE**
- MLE N-gram parameter estimation:

$$P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$$

- The above equation estimates the N-gram probability by dividing the observed frequency of a particular sequence by the observed frequency of a prefix.
- This ratio is called a relative frequency.

Bigram Approximation

- Bigram model **approximates** the probability of a word given all the previous words ***P(w_n / w_1^{n-1})*** **by using only the conditional probability of the preceding word *P(w_n / w_{n-1}).***
- Instead of computing the probability:

P(sugar| Mary likes her coffee with milk and)

we approximate it with the probability:

P(sugar/and)

- Thus, when we use a bigram model to predict the conditional probability of the next word we are thus making the following approximation:

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$$

Bigram probability

- To compute a bigram probability of a word **y** given a previous word **x**, we'll compute the count of the bigram xy, C(xy), and normalize by the sum of all the bigrams that share the same first word x:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)}$$

- Since the sum of all bigram counts that start with a given word w_{n-1} *must be equal to the unigram count for that word* w_{n-1}*, simplify*

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

Example : **mini-corpus of three sentences**

Sentence 1- I am Sam

Sentence 2- Sam I am

Sentence 3- I do not like green eggs and ham

- First, augment each sentence with a special symbol <s> at the beginning of the sentence, to give us the bigram context of the first word.
- Also, need a sentence end-symbol </s>.

1. <s> I am Sam </s>
2. <s> Sam I am </s>
3. <s> I do not like green eggs and ham </s>

contd…

- Calculations for some of the bigram probabilities from this corpus.

1. <s> I am Tom </s>
2. <s> Tom I am </s>
3. <s> I do not like green eggs and ham </s>
   - *P(I|<s>) = 2/3 =0* .67
   - *P(Tom|<s>) = 1/3=0* .33
   - *P(am|I) = 2/*3 = 0.67
   - *P(</s>|Tom) = 1/*2 = 0.5
   - *P(Tom|am) = 1/*2 = 0.5
   - *P(do|I) = 1/*3 = 0.33

contd …

- Consider the following probabilities:

  P(i|<s>) = 0.25 , **P(want|i)**=0.33,

  P(english|want)= 0.0011,

  P(food|english) = 0.5,

  P(</s>|food) = 0.68

- Compute the probability of sentences like **I want English food** or **I want Chinese food** by simply multiplying the appropriate bigram probabilities together, as follows:

Example:

P(<s> i want english food </s>) =

P(i|<s>) * P(want|i) * P(english|want) * P(food|english) * P(</s>|food)

= 0.25* 0.33 * 0.0011 * 0.50* 0.68

= **0.000031**

GPT-3 (Generative Pre-trained Transformer 3)

- One of the largest language model with 175 billion parameters, developed by OpenAI introduced in June 2020
- GPT-3 is built upon the Transformer architecture, a deep learning model
- It is known for its text generation capabilities and can be fine-tuned for various NLP tasks like language translation, summarization, text completion, and chatbot development , virtual assistants, content generation, sentiment analysis, and more.

BERT (Bidirectional Encoder Representations from Transformers)

- BERT is a transformer-based language model pretrained on large text corpora.
- Was introduced by Google AI researchers in 2018
- It captures bidirectional contextual information
- BERT is trained in 2 steps:
  - Pre-training
  - Fine tuning
- BERT is a deep model with multiple layers (usually 12 or 24) of bidirectional Transformer encoders.
- Used for tasks like text classification, question answering, named entity recognition and more.

ELMO (Embeddings from Language Models)

- Developed by researchers at the Allen Institute for Artificial Intelligence (AI2), was introduced in a paper titled "Deep Contextualized Word Representations" by Matthew E. Peters, et al in 2018 and was presented at EMNLP.
- Elmo introduced the concept of contextual word embeddings, which paved the way for later models like BERT and GPT to further advance natural language understanding and processing.

RoBERTa (A Robustly Optimized BERT Pretraining Approach)

- RoBERTa is a variant of BERT designed to improve training dynamics.
- It is pretrained on a massive text corpus and has achieved state-of-the-art results on various NLP benchmarks.

XLNet

- XLNet is another transformer-based model that extends the BERT model.
- It uses a permutation-based training approach to capture bidirectional context and has been successful in various NLP tasks.

T5

- T5 (Text-to-Text Transfer Transformer)
- T5 is a model that frames all NLP tasks as a text-to-text problem.
- It is pretrained on large-scale text data and fine-tuned for a wide range of NLP tasks, including translation and summarization.

DistilBERT

- DistilBERT is a distilled version of BERT, designed to be computationally efficient while maintaining good performance.
- It is used in applications where resource constraints are a concern.

Applications

- Speech Recognition
  - P(I saw a van) >> P(eyes awe of an)
- Machine Translation
  - Which sentence is more plausible in the target language?
    - P(**high winds) > P(large winds)**
- Context Sensitive Spelling Correction
- Natural Language Generation
- Completion Prediction

Smoothing Techniques

- What do we do with words that are in our vocabulary (they are not unknown words) but appear in a corpus in an unseen context.
- Removing off a bit of probability mass from some more frequent events and giving it to the events which have been never seen.
- This modification is called smoothing or discounting.

Smoothing Techniques

There are ways to do smoothing:

1. add-1 smoothing,
2. add-k smoothing,
3. Good Turing Smoothing, and
4. Kneser-Ney smoothing.

Laplace Smoothing or add-1 smoothing

- Add one to all the bigram counts, before normalizing them into probabilities.
- All the counts that used to be zero will now have a count of 1, the counts of 1 will be 2, and so on. This algorithm is called Laplace smoothing.

$$P^*_{\text{Laplace}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

where,

V is number of distinct words or vocabulary

	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

Figure 4.1 Bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences.

	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

Figure 4.5 Add-one smoothed bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences.

Unigram counts

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

	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

Figure 4.6 Add-one smoothed bigram probabilities for eight of the words (out of $V = 1446$) in the BeRP corpus of 9332 sentences.

Example 1

Consider the following corpus C of sentences:

1. **there is a big garden**
2. **children play in a garden**
3. **they play inside beautiful garden**

Calculate P(they play in a big garden) assuming a bi-gram language model?

Solution:

We have to Calculate: P(they play in a big garden)

Represent the sentence:

<s> there is a big garden </s>

<s> children play in a garden </s>

<s> they play inside beautiful garden </s>

P(they | <s>) = 1/3

P(play | they) = 1/1

P(in | play) = 1/2

P(a | in) = 1/1

P(big | a) = 1/2

P(garden | big) = 1/1

P(<\s>|garden) = 3/3

Sol:

P(they play in a big garden) =

$1/3 \times 1/1 \times 1/2 \times 1/1 \times 1/2 \times 1/1 \times 3/3 =$ **$1/12$**

Example 2:

Consider again the corpus C of sentences:

1. **there is a big garden**
2. **children play in a garden**
3. **they play inside beautiful garden**

Calculate **P(they play in a big garden)** assuming a bi-gram language model with add one smoothing.

Solution:

<s> there is a big garden </s>

<s> children play in a garden </s>

<s> they play inside beautiful garden </s>

Bigrams = {(there| <s>) , (is |there), (a|is), (big|a), (garden|big), (</s>|garden), (children|<s>), (play|children), (in|play),(a|in), (garden|a),(</s>|garden), (they|<s>), (play|they), (inside|play), (beautiful|inside), (garden|beautiful), (<\s>|garden) }

|V| = ??

Solution:

<s> there is a big garden </s>

<s> children play in a garden </s>

<s> they play inside beautiful garden </s>

Bigrams = {(there| <s>) , (is |there), (a|is), (big|a), (garden|big), (</s>|garden), (children|<s>), (play|children), (in|play),(a|in), (garden|a),(</s>|garden), (they|<s>), (play|they), (inside|play), (beautiful|inside), (garden|beautiful), (<\s>|garden) }

|V| = ??

contd...

$P(\text{they} \mid \text{<s>}) = (1+1) / (3+|V|)$

$P(\text{play} \mid \text{they}) = (1+1) / (1+|V|)$

$P(\text{in} \mid \text{play}) = (1+1) / (2+|V|)$

$P(\text{a} \mid \text{in}) = (1+1) / (1+|V|)$

$P(\text{big} \mid \text{a}) = (1+1) / (2+|V|)$

$P(\text{garden} \mid \text{big}) = (1+1) / (1+|V|)$

$P(<\backslash s> \mid \text{garden}) = (3+1) / (3+|V|)$

P(they play in a big garden) = ??

Add K smoothing

Good Turing Smoothing Technique

- **Basic Intuition**: Use the count of things we have seen once to help estimate the count of things we have never seen.
- For each count c, an adjusted count c* is computed as :

$$\mathbf{c^* = (c+1)\, N_{c+1} \,/\, N_c}$$

where,

N_c is the no of n-grams seen exactly c times

Contd...

Good Turning Smoothing:

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

where,

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

What if c=0,

$$P^*_{GT}\text{ (things with frequency c)} = N_1/N$$

where N denotes the total no of n-grams that actually occurs in training

Example

- Suppose you are reading an article, on Natural Language Processing. Till now, you have read the words: **language**-8 times, **aspect**-3 times, **processing**-2 times, **extraction**-2 times, **question**-once and **dialogue**-once.

1) What are the maximum likelihood estimate(MLE) probability ($P_{processing}$) and Good Turing probability ($P^*_{GT(processing)}$) for reading processing as the next word?

2) Calculate the MLE and Good Turing probabilities for reading "answering" as the next word?

Solution: Part 1

N_c = frequency of frequency c

language-8, question-1

aspect-3, dialogue-1

processing-2, extraction-2

N_1 =2 (no of unigrams with frequency count 1)

N_2 =2 (no of unigrams with frequency count 2)

N_3 =1 (no of unigrams with frequency count 3)

N_8 =1 (no of unigrams with frequency count 8)

$P_{processing}$ = count(processing) / count(vocab) = 2/17

$$P^*_{GT(processing)} = \frac{(c+1)\, N_{c+1} / N_c}{N}$$

$$P^*_{GT(processing)} = \frac{(2+1)\, N_3 / N_2}{17} = \frac{3*1/2}{17} = \frac{3}{34}$$

Solution: Part 2

N_c = frequency of frequency c

language-8, question-1

aspect-3, dialogue-1

processing-2, extraction-2

N_1 =2 (no of unigrams with frequency count 1)

N_2 =2 (no of unigrams with frequency count 2)

N_3 =1 (no of unigrams with frequency count 3)

N_8 =1 (no of unigrams with frequency count 8)

$P_{answering}$ = count(answering) / count(vocab) = 0/17 =0

$$P^*_{GT(answering)} = \frac{(c+1)\, N_{c+1} / N_c}{N}$$

$$P^*_{GT(answering)} = N_1 / N = \frac{2}{17}$$