Language Modeling/
N-gram language model

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

Introduction to N-gram

Language model :

- Probabilistic models (n-gram models)
- It is the task of predicting the next word in a text given the previous words / predict the next word from the previous N −1 words.
- Such statistical models of word sequences are also called **language models (LMs)**

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**

Computing P(W)

- How to compute the joint probability?

  P(about, fifteen, minutes, from)

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 $P(w_1, w_2, \ldots, w_n)$, using chain rule of probability :

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

contd…

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

- 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

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

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(sugar| 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

- The assumption that the probability of a word depends only on the previous word is called a **Markov assumption.**
- Markov models are the class of **probabilistic models** that assume that we can predict the probability of some future unit without looking too far into the past.
- We can generalize the **bigram** (which looks one word into the past) to the **trigram** (which looks two words into the past) and thus to the **N-gram (which looks *N –1 words into the past*).**
- Thus the **general equation for *N-gram approximation*** *to* the conditional probability of the next word in a sequence is:

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

contd…

- Given the bigram assumption for the probability of an individual word, we can compute the probability of a complete word sequence as:

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

- Now, how do we estimate these bigram or *N-gram probabilities?*

Probability Estimation for N-gram

- *The simplest and most* intuitive way to estimate probabilities is called **Maximum Likelihood Estimation, or MLE**
- The general case of **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.
  - $P(I|<s>) = 2/3 = 0.67$
  - $P(Sam|<s>) = 1/3 = 0.33$
  - $P(am|I) = 2/3 = 0.67$
  - $P(</s>|Sam) = 1/2 = 0.5$
  - $P(Sam|am) = 1/2 = 0.5$
  - $P(do|I) = 1/3 = 0.33$

Example:

- Compute the probability of sentences like **I want English food** using bigram langauge model?

Solution:

- **Given 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

  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**

Example 2:

- Consider the following corpus:

  Sentence 1 → I am Ram

  Sentence 2 → Ram I am

  Sentence 3 → I do not like brown bread and Jam

  Using bi-gram language model, calculate P(Ram do like bread and Jam) ?

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, 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

Solution:

P(they play in a big garden) =

1/3 x 1/1 x 1/2 x 1/1 x 1/2 x 1/1 x 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(they | <s>) = (1 +1) / (3 +|V|)

P(play | they) = (1 +1) / (1 +|V|)

P(in | play) = (1 +1) / (2 +|V|)

P(a | in) = (1 +1) / (1 +|V|)

P(big | a) = (1 +1) / (2 +|V|)

P(garden | big) = (1 +1) / (1 +|V|)

P(<\s>|garden) = (3 +1) / (3 +|V|)

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

Example 3:

Given a corpus C , the Maximum Likelihood Estimation (MLE) for the bigram dried berries is 0.3 and the count of occurrence of the word dried is 580. for the same corpus C, the likelihood of dried berries after applying add-one smoothing is 0.04. What is the vocabulary size of C ?

Sol:

$$P_{MLE}(berries|\ dried) = \frac{C(dried, berries)}{C(dried)}$$

$$0.3 = \frac{C(dried, berries)}{580}$$

$$C(dried, berries) = 174$$

$$P_{Add-1}(berries|\ dried) = \frac{C(dried, berries)+1}{C(dried)+ V}$$

$$0.04 = \frac{174+1}{580+V}$$

$$V = 3795$$

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 :

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

where,

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

N_c: Frequency of frequency c

Example Sentences::

<s>I am here </s>

<s>who am I </s>

<s>I would like </s>

N_c: Frequency of frequency c

Example Sentences::

<s> I am here </s>

<s> who am I </s>

<s> I would like </s>

Computing N_c

I	3
am	2
here	1
who	1
would	1
like	1

$N_1 = 4$

$N_2 = 1$

$N_3 = 1$

Good Turing Estimation

Idea:

- Reallocate the probability mass of n-grams that occur r+1 times in the training data to the n-grams that occur r times
- In particular, reallocate the probability mass of n-grams that were seen once to the n-grams that were never seen

Adjusted Count:

For each count c, an adjusted count c* is computed as :

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

where,

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

Good Turing Estimation

Good Turning Smoothing:

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

where,

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

What if c=0,

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

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

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

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

Points to remember

- Since probabilities are (by definition) less than or equal to 1, the more probabilities we multiply together, the smaller the product becomes.
- Multiplying enough N-grams together would result in numerical underflow.
- By using log probabilities instead of raw probabilities, we get numbers that are not as small.
- Adding in log space is equivalent to multiplying in linear space, so we combine log probabilities by adding them.
- The result of doing all computation and storage in log space is that we only need to convert back into probabilities if we need to report them at the end;
- then we can just take the exp of the log prob:

$$p1 \times p2 \times p3 \times p4 = \exp(\log p1 + \log p2 + \log p3 + \log p4)$$

Evaluation and Perplexity

Evaluation and Perplexity

- The probabilities of an N-gram model come from the corpus it is trained on, the training set or training corpus.
- The quality of an N-gram model is measured by its performance on some unseen data called the **test set or test corpus.**
- If we are given a corpus of text and want to compare two different N-gram models, we divide the data into training and test sets, train the parameters of models A and B on the training set, and then compare how well the two trained models model the test set.

What does it mean to "model the test set"?

- Whichever model assigns a higher probability to the test set—meaning it more accurately predicts the test set—is a better model.
- Given two probabilistic models, the better model is the one that has a tighter fit to the test data or that better predicts the details of the test data, and hence will assign a higher probability to the test data.

Points to remember

- Since our evaluation metric is based on test set probability, it's important not to let the test sentences into the training set.
- Suppose we are trying to compute the probability of a particular "test" sentence.
- If our test sentence is part of the training corpus, we will mistakenly assign it an artificially high probability when it occurs in the test set.
- We call this situation **training on the test set.**
- Training on the test set introduces a bias that makes the probabilities all look too high and causes huge inaccuracies in perplexity.
- In practice, we often just divide our data into 80% training, 10% development, and 10% test.

How do we evaluate models?

- Define an evaluation metric (scoring function).
  - We will want to measure how similar the predictions of the model are to real text.
- Train the model on a *'seen' training set.*
- Test the model on an *unseen test set.*
  - Test data must be disjoint from training data.

Perplexity

- Perplexity is the *de facto* standard for evaluating language models.
- We know, Language Modeling is the task of predicting the next word in a text given the previous words.
- For instance, given the history "Mary likes her coffee with milk and", a good language model might predict "sugar" and a bad language model might predict "socks."
- It's clear that there is **no "right answer"** to any of these prediction problem.

Perplexity

- Perplexity measures the cross entropy between the empirical distribution (the distribution of things that actually appear) and the predicted distribution (what your model likes) and then divides by the number of words and exponentiates after throwing out unseen words.

Entropy *H(X)*

- The **entropy *H(X) of a random variable X is the*** expected negative log probability:

$$H(X) = -\sum p(x) \log_2 p(x)$$

- Entropy is a measure of uncertainty.

The entropy of language

The entropy of sequences $W_1 \ldots W_n$:

$$H(W_1 W_2 \ldots W_n) = -\sum_{w_1 \ldots w_n \in \Sigma^n} p(w_1 \ldots w_n) \log_2 p(w_1 \ldots w_n)$$

The (per-word) entropy rate of sequences $W_1 \ldots W_n$:

$$\frac{1}{n} H(W_1 W_2 \ldots W_n) = -\frac{1}{n} \sum_{w_1 \ldots w_n \in \Sigma^n} p(w_1 \ldots w_n) \log_2 p(w_1 \ldots w_n)$$

The entropy of a language $L = \{W_1 \ldots W_n \mid 1 < n < \infty\}$

$$H(L) = -\lim_{n \to \infty} \frac{1}{n} H(W_1 \ldots W_n)$$

Perplexity

- The perplexity (sometimes called PP for short) of a language model on a test set is the inverse probability of the test set, normalized by the number of words.

- For a test set $W = w_1 w_2 \ldots w_N$,:

$$\begin{aligned} \mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}} \end{aligned}$$

contd …

- Use the chain rule to expand the probability of W:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

- If we are computing the perplexity of W with a bigram language model, we get:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

- The higher the conditional probability of the word sequence, the lower the perplexity.
- Thus, minimizing perplexity is equivalent to maximizing the test set probability according to the language model.

Perplexity

- Consider the task of recognizing the digits in English (zero, one, two,..., nine), given that each of the 10 digits occurs with equal probability P = 1/10 .
- The perplexity of this mini-language is in fact 10. To see that, imagine a string of digits of length N. The perplexity will be:

$$\begin{aligned} \text{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \left(\frac{1}{10}^{N}\right)^{-\frac{1}{N}} \\ &= \frac{1}{10}^{-1} \\ &= 10 \end{aligned}$$