**Probabilistic Language Modeling**

Create a language model from text corpus to

* estimate the probability of the word sequences
* estimate the probability of a word following the sequence of words

Apply this concept to autocomplete a sentence with most likely suggestions.



(image from deeplearning.ai)

**Other application of langue model**

1. **Speech recognition**
2. **Spelling Correction**
3. **Augmentative Communication**

**N-gram**

An N-gram is a sequence of N words

Corpus: I am happy because I am learning.

Unigrams: {I, am, happy, because, I, am, learning}

Bigrams: {I am, am happy, happy because, …}

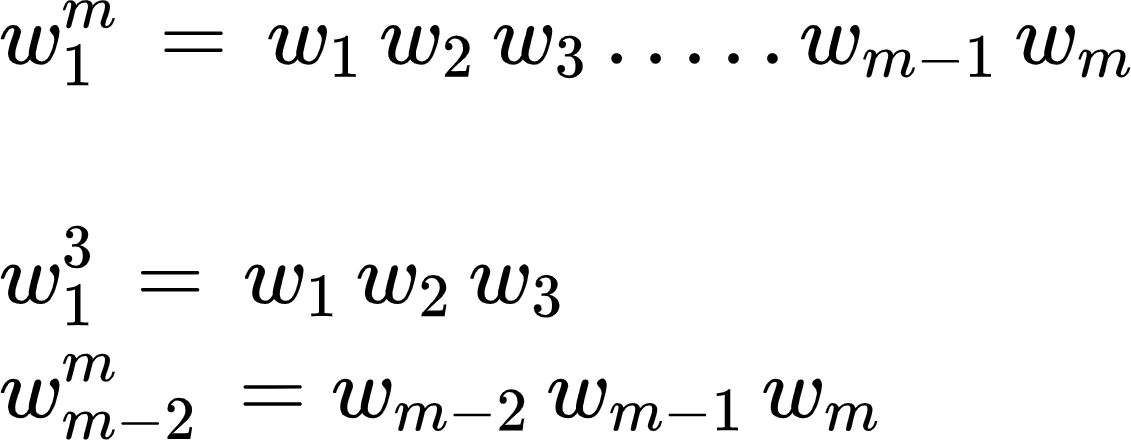
Trigrams: {I am happy, am happy because, …}

**Sequence Notation**

Corpus: This is great ……. teacher drinks tea.

here, w1 = This and so on..

let total no. of words, m = 500



**Unigram probability**

**Corpus:** I am happy because I am learning.

Size of corpus, m = 7

P(I) = 2/7 and P(happy) = 1/7

**Probability of unigram:** P(word) = C(word) / m

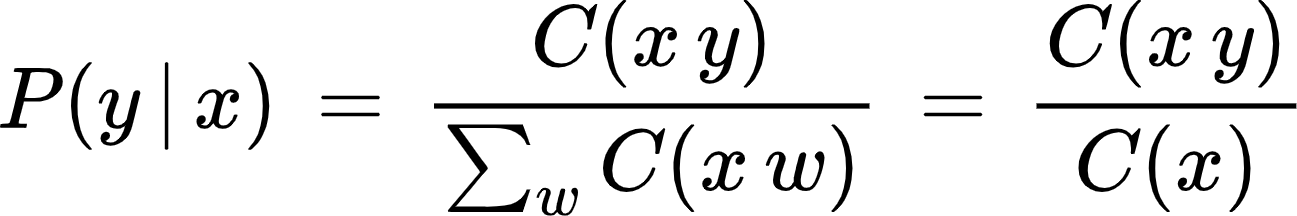
**Bigram Probability**

**Corpus:** I am happy because I am learning.

P(am | I) = C(I am) / C(I) = 2/2 = 1

P(happy|I) = C(I happy) / C(I) = 0 / 2 = 0

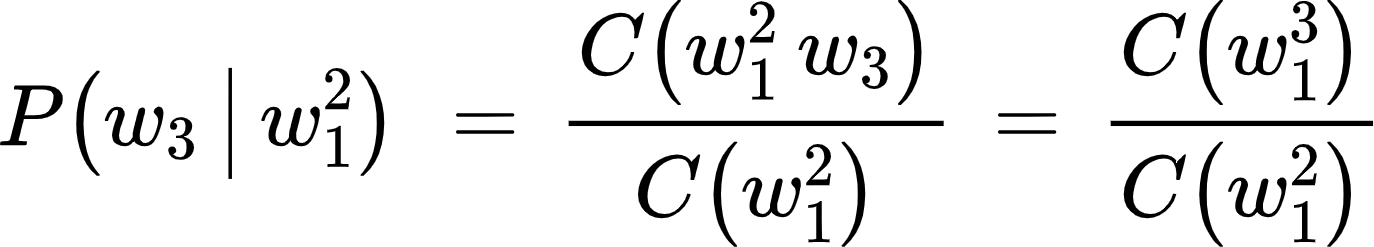
P(learning | am) = C(am learning) / C(am) = 1 / 2

**Probability of a bigram : **

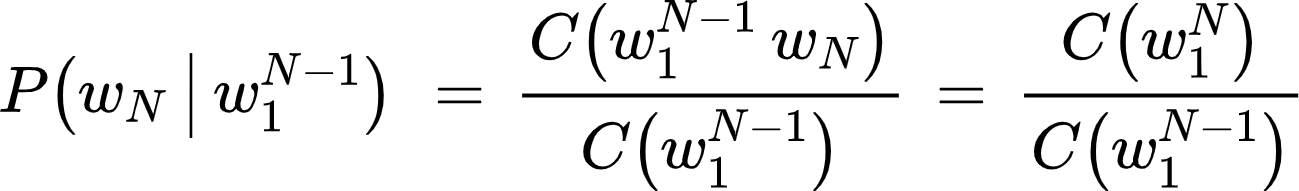
**Trigram Probability**

**Corpus:** I am happy because I am learning

P(happy | I am) = C(I am happy) / C(I am) = 1 / 2

**Probability of a trigram : **

**N-gram probability**

Probability of N-gram = 

**Probability of a sequence**

Given a sentence, what is its probability?

P(the teacher drinks tea) = ?

Conditional probability and chain rule reminder

P(B | A) = P(A B) / P(A) => P(A, B) = P(A) . (B | A)

P(A, B, C, D) = P(A) . P(B | A) . P(C | A, B) . P(D| A, B, C)

Hence

P(the teacher drinks tea) = P(the) . P(teacher | the) . P(drinks | the teacher) . P(tea | the teacher drinks)

**Problem:** Corpus almost never contains the exact sentence we’re interested in or even its longer subsequences.

Input: the teacher drinks tea

P(the teacher drinks tea) = P(the) . P(teacher | the) . P(drinks | the teacher) . P(tea | the teacher drinks)

here,

P(tea | the teacher drinks) = C(the teacher drinks tea) / C(the teacher drinks).

But the C(the teacher drinks tea) and C(the teacher drinks), both might be zero in the corpus.

Hence we can approximate them as

P(tea | the teacher drinks) P(tea } drinks)

hence, we have

P(the teacher drinks tea) = P(the) . P(teacher | the) . P(drinks | the teacher) . P(tea | the teacher drinks)

P(the teacher drinks tea) = P(the) . P(teacher | the) . P(drinks | teacher) . P(tea | drinks)

**Approximation of sequence probability**

**Markov Assumption: only last N words matter**

**Starting and Ending Process**

* start of the sentence symbols <s>
* end of the sentence symbols </s>

**Start of the sentence token <s>**

Input: the teacher drinks tea

P(the teacher drinks tea) P(the) P(teacher | the) P(drinks | teacher) P(tea | drinks)

Input: <s> the teacher drinks tea

P(the teacher drinks tea) P(the | <s>) P(teacher | the) P(drinks | teacher) P(tea | drinks)

In the case of trigrams:

Input: <s> <s> the teacher drinks tea

P(the teacher drinks tea) P(the | <s> <s>) P(teacher | <s> the) P(drinks | the teacher) P(tea | teacher drinks)

Similarly, we can write for n-gram

**End of the sentence token </s>**

Input: the teacher drinks tea

P(the teacher drinks tea) P(the) P(teacher | the) P(drinks | teacher) P(tea | drinks)

Input: <s> the teacher drinks tea </s>

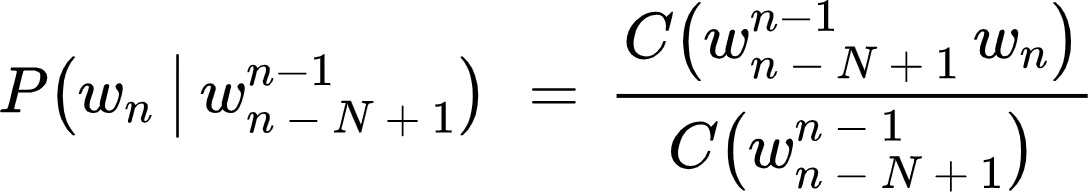
P(the teacher drinks tea) P(the | <s>) P(teacher | the) P(drinks | teacher) P(tea | drinks) P(</s> | tea)

**Outline**

* **Count matrix**
* **Probability matrix**
* **Language model**
* **Log probability to avoid underflow**
* **Generative language model**

**Count matrix**

Probability of the word after the sequence of word is given as:



Rows: unique corpus (N - 1) grams

Columns: unique corpus words

Here, we have bigram count matrix as follow

**Corpus:** <s> I study I learn </s>

|  | <s> | </s> | I | study | learn |
| --- | --- | --- | --- | --- | --- |
| <s> | 0 | 0 | 1 | 0 | 0 |
| </s> | 0 | 0 | 0 | 0 | 0 |
| I | 0 | 0 | 0 | 1 | 1 |
| study | 0 | 0 | 1 | 0 | 0 |
| learn | 0 | 1 | 0 | 0 | 0 |

**Probability matrix**

* Divide each cell by its row sum

|  | <s> | </s> | I | study | learn | sum |
| --- | --- | --- | --- | --- | --- | --- |
| <s> | 0 | 0 | 1 | 0 | 0 | 1 |
| </s> | 0 | 0 | 0 | 0 | 0 | 0 |
| I | 0 | 0 | 0 | 1 | 1 | 2 |
| study | 0 | 0 | 1 | 0 | 0 | 1 |
| learn | 0 | 1 | 0 | 0 | 0 | 1 |

We have probability matrix

|  | <s> | </s> | I | study | learn |
| --- | --- | --- | --- | --- | --- |
| <s> | 0 | 0 | 1 | 0 | 0 |
| </s> | 0 | 0 | 0 | 0 | 0 |
| I | 0 | 0 | 0 | 0.5 | 0.5 |
| study | 0 | 0 | 1 | 0 | 0 |
| learn | 0 | 1 | 0 | 0 | 0 |

**Language model**

From the probability matrix, we can calculate the sentence probability.

Input : <s> I learn </s>

P(input) = P(I | <s>) P(learn | I) P(</s> | learn) = 1 \* 0.5 \* 1 = 0.5

**Log Probability**

All probabilities in calculation <=1 and multiplying them brings risk of underflow.

**Generative Language model**

Corpus:

<s> Lyn drinks chocolate </s>

<s> John drinks tea </s>

<s> Lyn eats chocolate </s>

Language generation:

1. **(<s>, Lyn)** or (<s>, John)
2. (Lyn, eats) or **(Lyn, drinks)**
3. **(drinks, tea)** or (drinks, chocolate)
4. **(tea, </s>)**

Algorithm for the language generation:

Step 1: Choose sentence start

Step 2: Choose next bigram starting with the previous word

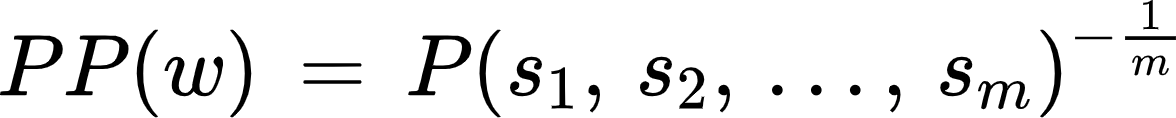
Step 3: Continue until </s> is picked

**Train test split**

**For large corpora:**

* 98% train
* 1% validation
* 1% test

**Perplexity**

****

**where,**

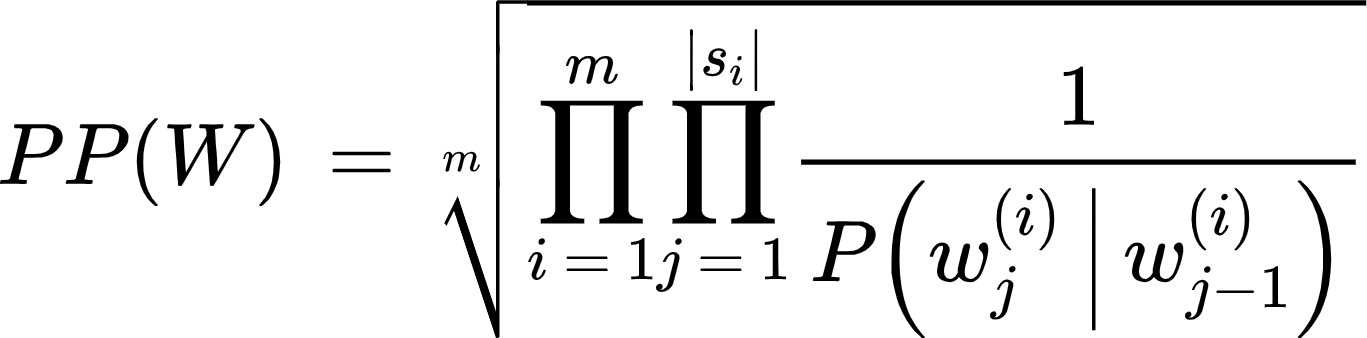
**w** ->test set containing m sentences s

**si**-> i-th sentence in the test set, each ending with </s>

**m** -> number of all words in entire test set W including </s> but not including <s>

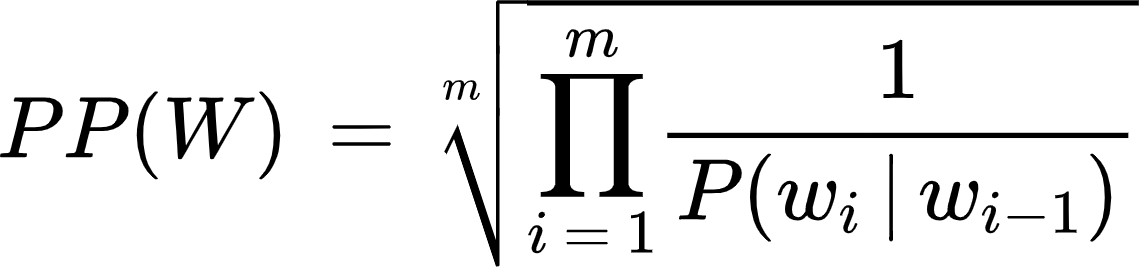
* Smaller perplexity => better model
* Character level models PP < word-based models PP

Perplexity for bigram models is given by



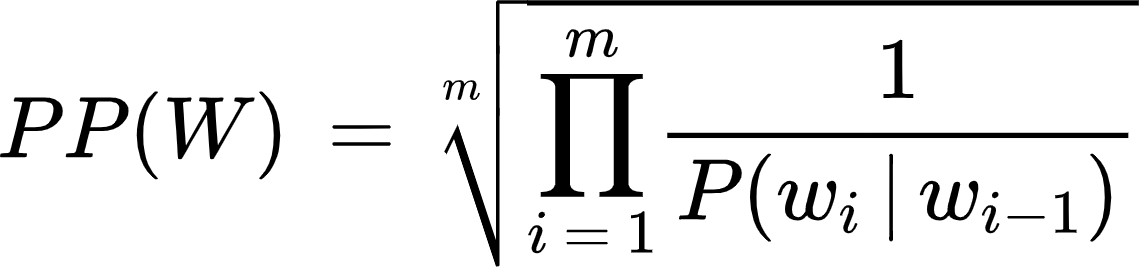
where, wj(i) -> j-th word in i-th sentence

Concatenate all sentences in W

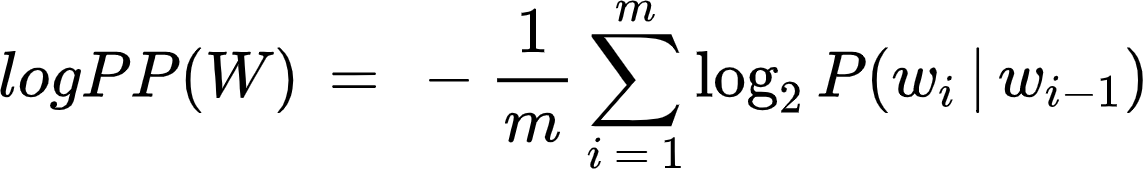


wi -> i-th word in test set

**Log Perplexity**



After taking log on both sides in above equation, we get



**Unknown words**

* Unknown words = Out of vocabulary words (OOV)
* special tag <UNK> in corpus and in input

**Using <UNK> in corpus**

* Create vocabulary V
* Replace any word in corpus and not in V by <UNK>
* Count the probabilities with <UNK> as with any other word

**Example**

Corpus:

<s> Lyn drinks chocolate </s>

<s> John drinks tea </s>

<s> Lyn eats chocolate </s>

If the minimum frequency to create vocabulary = 2, then the required corpus is as follow

<s> Lyn drinks chocolate </s>

<s> <UNK> drinks <UNK> </s>

<s> Lyn <UNK> chocolate </s>

So, we have vocabulary as follow

Lyn, drinks, chocolate

**Input query :** <s> Adam drinks chocolate </s>

<s> <UNK> drinks chocolate </s>

**How to create vocabulary V**

* **Criteria**
  + Min word frequency f
  + Max |V|, include words by frequency
* Use <UNK> sparingly
* Perplexity - only comparees LMs with the same V

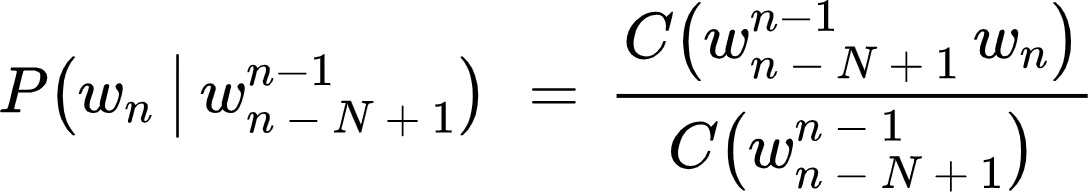
**Outline**

* Missing N-grams in corpus
* Smoothing
* Backoff and interpolation

**Missing N-grams in corpus**

Problem: N-grams made of known words still might be missing in the training corpus.

Their counts cannot be used for probability estimation

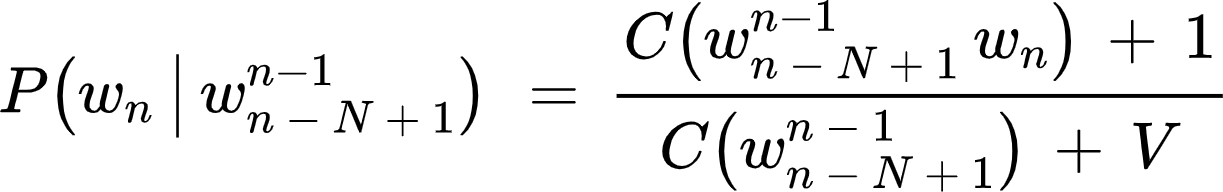


Here, both the count in numerator and denominator can be zero and hence probability may be either zero or infinite or undefined.

**Smoothing**

Smoothing solves the problem mentioned above due to missing N-grams in corpus.

1. Laplacian smoothing



1. Add-k smoothing

