WEB INTELLIGENCE

Lecture 5: Language Modeling I

Russa Biswas

October 14, 2025

AALBORG UNIVERSITY

(Based on slides of Dan Jurafsky)

Probabilistic Language Modeling

- Today's goal: assign a probability to a sentence
  - Machine Translation:
    - P(**high** winds tonight) > P(**large** winds tonight)
  - Spell Correction
    - The office is about fifteen **minuets** from my house
    - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
  - Speech Recognition
    - P(I saw a van) >> P(eyes awe of an)
  - + Summarization, question-answering, etc., etc.!!

Probabilistic Language Modeling

- Goal: compute the probability of a sentence or sequence of words:

  $P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$

- Related task: probability of an upcoming word:

  $P(w_5|w_1,w_2,w_3,w_4)$

- A model that computes either of these:

  $P(W)$ or $P(w_n|w_1,w_2 \ldots w_{n-1})$ is called a **language model**.

- Better: **the grammar** But **language model** or **LM** is standard

How to compute P(W)?

How to compute P(W)?

- How to compute this joint probability:
  - P(its, water, is, so, transparent, that)
- Intuition: let's rely on the Chain Rule of Probability

- Definition of conditional probabilities

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

Rewriting: $P(A \cap B) = P(A)P(B|A)$ or $P(A,B) = P(A)P(B|A)$

- More variables:
  - P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)
- The Chain Rule in General
  - $P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$

If A and B are independent

$$P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{P(A)P(B)}{P(A)} = \mathrm{P(B)}$$

Chain Rule to Compute the Joint Probability

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

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water) × P(so|its water is) × P(transparent|its water is so)

How to estimate these probabilities?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

Limitations:

- No! Too many possible sentences!
- We'll never see enough data for estimating these

Markov Assumption

- Simplifying assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

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

In other words, we approximate each component in the product

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

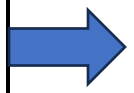
Unigram Model

- Most simplest model
- Consider a sentence: *“I love programming”*
  - P(“I love programming") = P("I") × P("love") × P("programming")
- Probability of each word is assumed to depend **only on itself and not** on the preceding words
- Hence, the probability of a word appearing in a sequence is **independent** of the words that came before it.

```
fifth, an, of, futures, the, an, incorporated, a,
a, the, inflation, most, dollars, quarter, in, is,
mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the
```



Some automatically generated sentences from a unigram model

Bigram Model

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

Consider the sentence: *"I love programming"*

P("I love programming") = P("I") × P("love"|"I") × P("programming"|"love")

Every word in the sequence has a probability that is solely dependent on its preceding word, and not on any other earlier words in the sequence.

N-grams Model

- We can extend to trigrams, 4-grams, 5-grams

Trigram: $$P(S) = \prod_{i=1}^{n} P(w_i|w_{i-1}, w_{i-2})$$

N-gram: $$P(S) = \prod_{i=1}^{n} P(w_i|w_1, \cdots, w_{i-1})$$

- In general this is an insufficient model of language because language has **long-distance dependencies**:
  - "The computer which I had just put into the machine room on the fifth floor crashed."
  - "The computer which I had just put into the machine room on the fifth **floor** **crashed**."
  - "The **computer** which I had just put into the machine room on the fifth **floor** **crashed**."

- But we can often get away with N-gram models
- Space Complexity: $O(|V|^n)$,
  - where V = vocabulary, n = max. length of sentence
    - E.g.: 475,000 main headwords in Webster's Third New International Dictionary
    - Average English sentence length: 14.3 words
    - A rough estimate: $O(475{,}00014) \sim 3.38 \cdot 1066$ TB
- By applying an N-gram model (say, N =4), we make the model more compact: $O(475{,}000^4)$

Estimating N-Gram Probabilities

Bi-grams:

The **Maximum Likelihood Estimate**

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

Example:

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

Vocabulary: {I, am, Sam, do, not, like, green, eggs, and, ham}

Maximum Likelihood Estimation

	I	am	Sam	do	not	like	green	eggs	and	ham	<s>	</s>
I	0	2	0	1	0	0	0	0	0	0	0	0
am	0	0	1	0	0	0	0	0	0	0	0	1
Sam	1	0	0	0	0	0	0	0	0	0	0	1
do	0	0	0	0	1	0	0	0	0	0	0	0
not	0	0	0	0	0	1	0	0	0	0	0	0
like	0	0	0	0	0	0	1	0	0	0	0	0
green	0	0	0	0	0	0	0	1	0	0	0	0
eggs	0	0	0	0	0	0	0	0	1	0	0	0
and	0	0	0	0	0	0	0	0	0	1	0	0
ham	0	0	0	0	0	0	0	0	0	0	0	1
<s>	2	0	1	0	0	0	0	0	0	0	0	0
</s>	0	0	0	0	0	0	0	0	0	0	0	0

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

Vocabulary: {I, am, Sam, do, not, like, green, eggs, and, ham}

Maximum Likelihood Estimation

$$P(\texttt{I}\,|\,\texttt{<s>}) = \frac{2}{3} = .67 \qquad P(\texttt{Sam}\,|\,\texttt{<s>}) = \frac{1}{3} = .33 \qquad P(\texttt{am}\,|\,\texttt{I}) = \frac{2}{3} = .67$$

$$P(\texttt{</s>}\,|\,\texttt{Sam}) = \frac{1}{2} = 0.5 \qquad P(\texttt{Sam}\,|\,\texttt{am}) = \frac{1}{2} = .5 \qquad P(\texttt{do}\,|\,\texttt{I}) = \frac{1}{3} = .33$$

Estimation of the Maximum Likelihood Estimation of the sentence:

<s> I am Sam </s>

P(S) = P (I | <s>)× P (am | I)× P (Sam | am) × P (</s> | Sam)

= 0.67 x 0.67 x 0.5 x 0.5 = 0.112

Maximum Likelihood Estimation

- What if a word occurs that is not part of our vocabulary?
  - “I am John”
- Closed Vocabulary Assumption: The test set can contain only words from our vocabulary.
- Open Vocabulary Assumption: The test set can contain unknown words (out of vocabulary words, OOV) that are not part of our vocabulary.
- In an Open Vocabulary system, unknown words are modeled by adding a pseudo-word <UNK>
  - Convert in the training set any word not in the vocabulary (OOV word) into <UNK>
  - Convert in the test set any unknown word into <UNK>
  - Estimate probabilities for <UNK> like for any regular vocabulary word.

Maximum Likelihood Estimation

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

Find the probability of the sentence

<s> I am John </s>

P(S) = P (I | <s>) x P (am | I) x P (John | am) x P (</s> | John)

= 0.67 x 0.67 x 0 x 0 = 0

Maximum Likelihood Estimation

1. I love machine learning
2. I love deep learning
3. Deep learning is great
4. Machine learning is fun

Calculate MLE for bi-grams.

Maximum Likelihood Estimation

Example:

- Shakespeare corpus consists of N=884,647 word tokens and a vocabulary of V=29,066 word types.
- Only 30,000 word types occurred (of possible 475,000 referenced by Webster's 3rd New International Dictionary).
  - Words not in the training data ⇒ *P(unknown Word)=0*
- Only **0.04% of all possible 2-grams** occurred.
- The probability of **99.96% of all possible 2-grams is 0**.

Maximum Likelihood Estimation

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

P(I|<s>) × P(want|I) × P(english|want) × P(food|english) × P(</s>|food)

= .000031

What kind of knowledge?

- P(english|want) = .0011
- P(chinese|want) = .0065
- P(to|want) = .66
- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0
- P (I | <s>) = .25

Maximum Likelihood Estimation

- We do everything in log space
  - Avoid underflow
  - adding is faster than multiplying

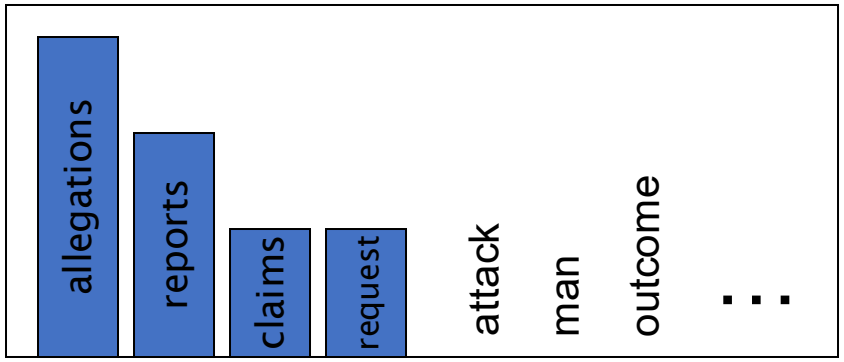
$$\log(p_1 \square p_2 \square p_3 \square p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Smoothing

The intuition of smoothing (from Dan Klein)

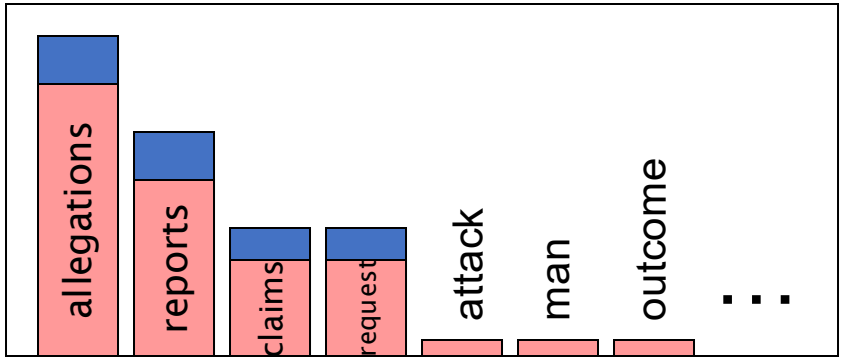
- When we have sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



- Steal probability mass to generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



Smoothing

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did
- Just add one to all the counts!

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

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

Berkeley Restaurant Corpus: Laplace smoothed bigram counts

	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	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

Laplace Smoothed Bigram Counts

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

	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

Reconstituted Counts

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

	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

Smoothing in N-grams

	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

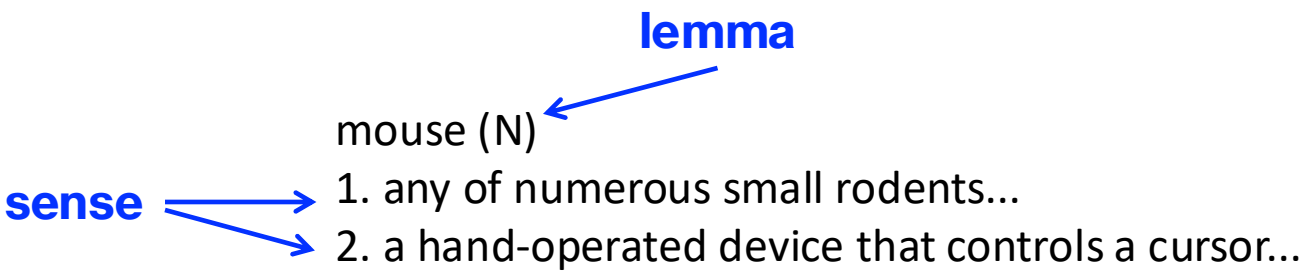
	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	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

- So add-1 isn't used for N-grams:
  - We'll see better methods
- But add-1 is used to smooth other NLP models
  - For text classification
  - In domains where the number of zeros isn't so huge

Word Meaning

- N-gram methods we've seen so far
  - Words are just strings (or indices w_i in a vocabulary list)
  - That's not very satisfactory!
- Using Logics
  - The meaning of "dog" is DOG; cat is CAT

$$\forall x \; DOG(x) \longrightarrow MAMMAL(x)$$



A sense or "concept" is the meaning component of a word
Lemmas can be polysemous (have multiple senses)

Word Meaning: Synonyms

- Synonyms have the same meaning in some or all contexts.
  - filbert / hazelnut
  - couch / sofa
  - big / large
  - automobile / car
  - water / H_20
- there are probably no examples of perfect synonymy.
  - Even if many aspects of meaning are identical
  - big/large

    my big sister != my large sister

Ask Humans?

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

SimLex-999 dataset (Hill et al., 2015) gives values on a scale from 0 to 10, from near-synonyms (*vanish*, *disappear*) to pairs that scarcely seem to have anything in common (*hole*, *agreement*)

Word Relatedness

Also called "word association"
Words can be related in any way, perhaps via a semantic frame or field

coffee, tea: **similar**
coffee, cup: **related**, not similar

- Words that
  - cover a particular semantic domain
  - bear structured relations with each other.

hospitals

surgeon, *scalpel*, *nurse*, *anaesthetic*, *hospital*

restaurants

waiter, *menu*, *plate*, *food*, *menu, chef*

houses

door, *roof*, *kitchen*, *family*, *bed*

Computational models of word meaning

Vector semantics

- The standard model in language processing!
- Handles many of our goals!
- Each word = a vector (not just "good" or "w_{45}")
  - Similar words are "**nearby in semantic space**"
  - We build this space automatically by seeing which words are **nearby in text**

Literature

- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press. 2008.
- https://stanford-cs324.github.io/winter2022/lectures/introduction/