Language modelling

February 8, 2021

Overview

"You are uniformly charming!" cried he, with a smile of associating and now and then I bowed and they perceived a chaise and four to wish for.

–Random sentence generated from a Jane Austen trigram model
Materials of this lecture slides are from the following book, not our IIR book.
Chapter 3: N-gram Language Models, Speech and Language Processing.
Daniel Jurafsky, James H. Martin. 2019.

Probabilistic Language Models

- What is Language Model (LM): model that assigns a probability to a sequence of words
- Applications
  - Machine Translation:

$$P(\mathit{high\ winds\ tonite}) > P(\mathit{large\ winds\ tonite}) \tag{1}$$

  - Spell Correction

$$\mathit{The\ office\ is\ about\ fifteen\ minuets\ from\ my\ house} \tag{2}$$

$$P(\mathit{about\ fifteen\ minutes\ from}) > P(\mathit{about\ fifteen\ minuets\ from}) \tag{3}$$

  - Speech Recognition

$$P(\mathit{I\ saw\ a\ van}) \gg P(\mathit{eyes\ awe\ ofan}) \tag{4}$$

  - Text summarization, question-answering, ...

Probabilistic Language Modeling

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

$$P(S) = P(w_1, w_2, w_3, w_4, w_5, \ldots, w_n) \tag{5}$$

- Related task: probability of an upcoming word:

$$P(w_5 | w_1, w_2, w_3, w_4) \tag{6}$$

- A model that computes either $P(S)$ 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(S)

- How to compute this joint probability:

$$P(its, water, is, so, transparent, that)$$

- Intuition: let's rely on the Chain Rule of Probability

The chain rule

- Conditional probabilities

$$P(A, B) = P(A\ and\ B) = P(A)P(B|A) = P(B)P(A|B) \tag{7}$$

- More variables:

$$P(A, B, C, D) = P(A)P(B|A)P(C|A, B)P(D|A, B, C) \tag{8}$$

- The Chain Rule in General

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

- For a sentence:

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

-

$$\begin{aligned} P(its\ water\ is\ so\ transparent) &= P(its) \\ &\times P(water|its) \\ &\times P(is|its\ water) \\ &\times P(so|its\ water\ is) \\ &\times P(transparent|its\ water\ is\ so) \end{aligned} \tag{9}$$

- Count and divide?

$$P(\mathit{the}|\mathit{its\ water\ is\ so\ transparent\ that}) \tag{10}$$

$$= \frac{\mathit{count}(\mathit{its\ water\ is\ so\ transparent\ that\ the})}{\mathit{count}(\mathit{its\ water\ is\ so\ transparent\ that})} \tag{11}$$

- Too many possible sentences
- Not enough data for estimating

- predict the probability of a sequence of words.
- n-gram language model:

$$p(w_1, w_2, \ldots, w_T) = \prod_i p(w_i | w_{i-1}, \ldots, w_{i-n+1}) \tag{12}$$

- Derived using chain rule and Markov assumption.

$$p(w_t | w_{t-n+1}, \ldots, w_{t-1}) = \frac{Count(w_{t-n+1}, \ldots, w_t)}{Count(w_{t-n+1}, \ldots, w_{t-1})} \tag{13}$$

- The problem of this approach: scarcity of data

Markov assumption

- Simplifying assumption:

$$P(\textit{the}|\textit{its water is so transparent that}) \approx P(\textit{the}|\textit{that})$$

- Or maybe

$$P(\textit{the}|\textit{its water is so transparent that}) \approx P(\textit{the}|\textit{transparent that})$$

- Markov assumption

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

In other words, we approximate each component in the product

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx \prod_i P(w_i | w_{i-k} \dots w_{i-1})$$

Simplest case: the unigram model

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$$

Some automatically generated sentences from a unigram model:

- 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

bigram model

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

Some automatically generated sentences from a bigram model:

```
texaco rose one in this issue is pursuing growth
in a boiler house said mr. gurria mexico 's motion
control proposal without permission from five
hundred fifty five yen

outside new car parking lot of the agreement
reached

this would be a record november
```

- We can extend to trigrams, 4-grams, 5-grams
- 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."
- But we can often get away with N-gram models

Estimating bigram probabilities

The Maximum Likelihood Estimate

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

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

$P(I| < s >) = \frac{2}{3}$ $P(am|I) = \frac{2}{3}$ $P(Sam|am) = \frac{1}{2}$ $P(< /s > |Sam) = \frac{1}{2}$

More examples: Berkeley Restaurant Project sentences

- Example sentences

```
can you tell me about any good cantonese restaurants  close by
mid priced thai food is what 'im looking for
tell me about chez panisse
can you give me a listing of the kinds of food that are availa
im looking for a good place to eat breakfast
when is caffe venezia open during the day
```

Raw bigram counts

Out of 9222 sentences

	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

Normalized bigram

unigram:

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

normalized by the unigram:

	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

e.g., i want=827/2533=0.33

Bigram estimates of sentence probability

$$
\begin{aligned}
&P(< s > I\ want\ english\ food\ < /s >)\\
&= P(I| < s >)\\
&\times P(want|I)\\
&\times P(english|want)\\
&\times P(food|english)\\
&\times P(< /s > |food)\\
&= .000031
\end{aligned}
$$

$$
\begin{aligned}
P(i| < s >) &= .25\\
P(english|want) &= .0011\\
P(chinese|want) &= .0065\\
P(to|want) &= .66\\
P(eat|to) &= .28\\
P(food|to) &= 0\\
P(want|spend) &= 0
\end{aligned}
$$

Practical issues

We do everything in log space

- Avoid underflow
- also adding is faster than multiplying

$$\log(p_1 \times p_2) = \log(p_1) + \log(p_2)$$

Google n-gram

```
serve as the incoming 92
serve as the incubator 99
serve as the independent 794
serve as the index 223
serve as the indication 72
serve as the indicator 120
serve as the indicators 45
serve as the indispensable 111
serve as the indispensible 40
serve as the individual 234
```

google book ngram: http://ngrams.googlelabs.com/
http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

Perplexity

The Shannon Game:

- How well can we predict the next word?
  - I always order pizza with cheese and mushrooms 0.1

pepperoni 0.1

anchovies 0.01

...

fried rice 0.0001

...

and 1e-100

The 33rd President of the US was ___
I saw a __

- Unigrams won't work for this task.
- A better model of a text is one which assigns a higher probability to the word that actually occurs

Perplexity

- The best language model is one that best predicts an unseen test set (Gives the highest P(sentence))
- Perplexity is the inverse probability of the test set, normalized by the number of words:

$$PP(S) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

- Chain rule:

$$PP(S) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 w_2 \ldots w_{i-1})}}$$

- For bigrams:

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

The Shannon Game intuition for perplexity

- How hard is the task of recognizing digits '0,1,2,3,4,5,6,7,8,9'
  - Let's suppose a sentence consisting of random digits
  - What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

$$
\begin{aligned}
PP(S) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \left[\left(\frac{1}{10}\right)^N\right]^{-\frac{1}{N}} \\
&= \left(\frac{1}{10}\right)^{-1} \\
&= 10
\end{aligned} \tag{14}
$$

- How hard is recognizing (30,000) names at Microsoft.
  - Perplexity = 30,000
- If a system has to recognize
  - Operator (1 in 4)
  - Sales (1 in 4)
  - Technical Support (1 in 4)
  - 30,000 names (1 in 120,000 each)
  - Perplexity is 53

$$
\begin{aligned}
&(\frac{1}{4}^{30k} \times \frac{1}{4}^{30k} \times \frac{1}{4}^{30k} \times \frac{1}{120k}^{30k})^{\frac{-1}{120k}} && (15)\\
&= (4 * 4 * 4 * 120000)^{1/4} && (16)\\
&\approx 53 && (17)
\end{aligned}
$$

- Perplexity is weighted equivalent branching factor

Lower perplexity = better model

- Training 38 million words, test 1.5 million words, WSJ

	Unigram	Bigram	Trigram
Perplexity	962	170	109

Approximating Shakespeare

Unigram

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
Every enter now severally so, let
Hill he late speaks; or! a more to leg less first you enter
Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

Bigram

What means, sir. I confess she? then all sorts, he is trim, captain.
Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

Trigram

Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
This shall forbid it should be branded, if renown made it empty.
Indeed the duke; and had a very good friend.
Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

Quadrigram

King Henry.What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'
Will you not tell me who I am?
It cannot be but so.
Indeed the short and the long. Marry, 'tis a noble Lepidus.

Shakespeare as corpus

- N=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of V2= 844 million possible bigrams.
- So 99.96% of the possible bigrams were never seen (have zero entries in the table)
- Quadrigrams worse: What's coming out looks like Shakespeare because it is Shakespeare

The wall street journal is not shakespeare

Unigram

Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives

Bigram

Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her

Trigram

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Using language models (LMs) for IR

1. LM = language model
2. We view the document as a generative model that generates the query.
3. What we need to do:
   - Define the precise generative model we want to use
   - Estimate parameters (different parameters for each document's model)
   - Smooth to avoid zeros
   - Apply to query and find document most likely to have generated the query
   - Present most likely document(s) to user
4. Note that 3 is similar to what we did in Naive Bayes.

Zeros and Smoothing

- Training set:
  - … denied the allegations
  - … denied the reports
  - … denied the claims
  - … denied the request
- Test set
  - … denied the offer
  - … denied the loan
- $P(offer|denied\ the) = 0$

Takeaways

- Daniel Jurafsky & James H. Martin, N-gram Language Models, book chapter.
- IIR, Chapter 12, p218-264.