N-gram Language Model

Speech and Language Processing Ch. 3

https://web.stanford.edu/~jurafsky/slp3/3.pdf

“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

Language Modeling

- Is to **assign a probability** to each possible next **word** or to an **entire sentence**
- E.g., *You say good…* bye > *die*
- “all of a sudden I notice three guys standing on the sidewalk.” >

  “on guys all I of notice sidewalk three a sudden standing the”

Applications

- Machine Translation:
  - P(**high** winds tonite) > P(**large** winds tonite)
- 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:

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

- Related task: probability of an upcoming word:

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

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

The Chain Rule

- Conditional probabilities

$$p(B|A) = P(A,B)/P(A) \quad \Rightarrow \quad P(A,B) = P(A)P(B|A)$$

- More variables:

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

- Chain Rule in General

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

Compute joint probability of words in sentence

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

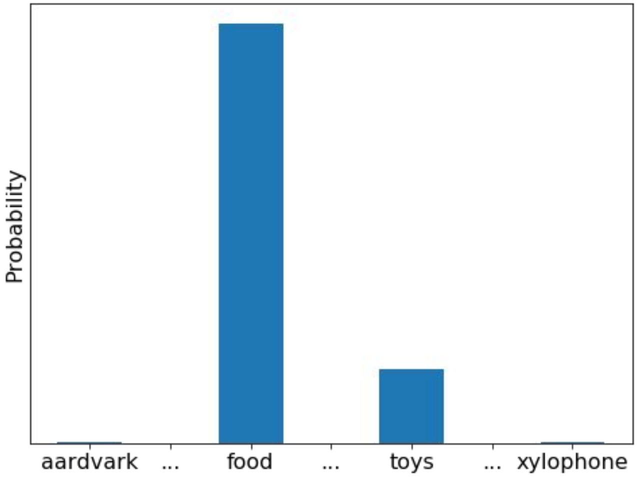
P("its water is so transparent") = P(its) × P(water|its) × P(is|its water) × P(so|its water is) × P(transparent|its water is so)

Distribution of the next words' probabilities

$$p(w_n \mid w_1, \ldots, w_{n-1})$$

Guess the word for ___ with given

[my, cat's, breath, smells, like, cat, ____]



https://medium.com/phrasee/neural-text-generation-generating-text-using-conditional-language-models-a37b69c7cd4b

Choose the next word

- **Sampling**: Sample from the conditional word probability distribution. Words that are a better fit are more likely to be selected
  - select “food” with probability 62%, “toys” with probability 14%, “aardvark” with probability 0.001%, etc.
- **Greedy**: Always pick the word with the highest probability - “food”.
- **Beam search**: greedy approach doesn’t always result in the final sequence with the highest overall probability. A beam search keeps track of several probable variants at each step to avoid being led astray by local maxim
  - Select “food” and “toys”, and reassess what is better when more words have been added.

https://medium.com/phrasee/neural-text-generation-generating-text-using-conditional-language-models-a37b69c7cd4b

Estimate these probabilities

- Could we just count and divide?

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

- Can you compute ***P(its water is so transparent)*** *?*
- No! Too many possible sentences!
- We'll never see enough data for estimating these
- Language is creative and any particular sentence may have never occurred

"Walden Pond's water is so transparent that the"

Approximate it

- We need cleverer ways of estimating the probability of a word *w* given a history *h*
- Instead of computing the probability of a word *w* given its entire history *h*
- **Approximate the history by the last few words**!

Markov Assumption

- The probability of a word depends only on the previous word

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

- Or

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

Markov Assumption

Generalize the bigram to n-gram (which looks n-1 words into the past)

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

앞 k개 단어로 결정

Markov Assumption

Generalize the bigram to n-gram (which looks n-1 words into the past)

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

앞 k개 단어로 결정

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

Markov Assumption

Generalize the bigram to n-gram (which looks n-1 words into the past)

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

앞 k개 단어로 결정

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

approximate each component in the product

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

Markov Assumption

Generalize the bigram to n-gram (which looks n-1 words into the past)

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

앞 k개 단어로 결정

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

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

E.g., *P(I do not like green eggs and ham)*

Bigram model

Condition on the previous word:

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

```
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
```

N-gram models

- Extending to trigrams, 4-grams, 5-grams
- is generally an insufficient model because **language has long-distance dependencies**:

"The **computer**(s) which I had just put into the machine room on the fifth floor **is** (are) crashing."

- But we can often get away with N-gram models
- How do we estimate these bigram or n-gram probabilities?

Estimating bigram probabilities

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

Example

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

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

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

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

Examples: Berkeley Restaurant Project sentences (Jurafsky et al., 1994)

- A dialogue system that answered questions about a database of restaurants in Berkeley, California
- 9,222 sentences
- Sample nomalized user queries
    - can you tell me about any good cantonese restaurants close by
    - mid priced thai food is what i’m looking for
    - tell me about chez panisse
    - can you give me a listing of the kinds of food that are available
    - i’m looking for a good place to eat breakfast
    - when is caffe venezia open during the day

Raw counts

Unigram

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

Bigram

	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

Bigram estimates of sentence probabilities

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

Given the followings

P(i|<s>) = 0.25

P(english|want) = 0.0011

P(food|english) = 0.5

P(</s>|food) = 0.68

Bigram estimates of sentence probabilities

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

P(I|<s>)

× **P(want|I)**

× P(english|want)

× P(food|english)

× P(</s>|food)

Given the followings

P(i|<s>) = 0.25

P(english|want) = 0.0011

P(food|english) = 0.5

P(</s>|food) = 0.68

Bigram estimates of sentence probabilities

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

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .25×.33×.0011×0.5×0.68

= .000031

P(i|<s>) = 0.25

P(english|want) = 0.0011

P(food|english) = 0.5

P(</s>|food) = 0.68

	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

Bigram estimates of sentence probabilities

P(<s> I want chinese food </s>)

P(i|<s>) = 0.25
P(english|want) = 0.0011
P(food|english) = 0.5
P(</s>|food) = 0.68

	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

Practical Issues

- common to use trigram models, which condition on the previous two words rather than the previous word, or 4-gram or even 5-gram models, when there is sufficient training data.
- We do everything in log space
  - Avoid underflow
  - Adding is faster than multiplying

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

Summary

- Language model computes the possibility of a sentence or an upcoming word
- Estimate the model using Markov Assumption
- Generalization via smoothing