Language Modeling, N-Gram Models

using examples from the text Jurafsky and Martin, and from slides by Dan Jurafsky

Language Models

- The goal of a Language Model is to assign a probability that a sentence (or phrase) will occur in natural uses of the language
- Why?
    - 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, and many other NLP applications

Language Models

- 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:

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

  conditional probability that w_5 occurs, given that we know that w_1,w_2,w_3,w_4 already occurred.

- 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**.

- We might call this a grammar because it predicts the (word-level) structure of the language, but language model is the standard terminology.

Chain rule applied

Compute the probability of a sentence by computing the joint probability of all the words conditioned by the previous words

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

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)

Computing Probabilities

- Normally, we just compute the probability that something occurred by counting its occurrences and dividing by the total number

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

- But there are way too many unique English sentences in any realistic corpus for this to work!
  - We'll never see enough data

Reminder: Conditional Probability: P(A|B) = P(A,B)(joint probability) / P(B)

Markov Assumption


Andrei Markov

- Instead we make the simplifying Markov assumption that we can predict the next word based on only one word previous:

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

- Or perhaps two words previous:

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

N-gram models

- Unigram Model: (word frequencies)
  The simplest case is that we predict a sentence probability just based on the probabilities of the words with no preceding words

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

- Bigram Model: (two word frequencies)
  Prediction based on one previous word:

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

Bigrams

- Examples of bigrams are any two words that occur together
  - In the text: "two great and powerful groups of nations", the bigrams are "two great", "great and", "and powerful", etc.
- **The *frequency* of an n-gram** is the percentage of times the n-gram occurs in all the n-grams of the corpus and could be useful in corpus statistics
  - For bigram xy:
    - Count of bigram xy / Count of all bigrams in corpus
  - Examples are in the Google N-gram corpus
- But in bigram language models, we use **the *bigram probability*, meaning a conditional probability,** to predict how likely it is that the second word follows the first

N-gram Models

- We can extend to trigrams, 4-grams, 5-grams
  - Each higher number will get a more accurate model, but will be harder to find examples of the longer word sequences in the corpus
- 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 last word *crashed* is not very likely to follow the word *floor*, but it is likely to be the main verb of the word *computer*

- But we can often get away with N-gram models

N-Gram probabilities

- For N-Grams, we need the conditional probability:

  P(<next word> | <preceding word sequence of length n>)

  e.g. P (*the* | *They picnicked by*)

- We define this as
  - the observed frequency (count) of the whole sequence divided by
  - the observed frequency of the preceding, or initial, sequence (sometimes called the maximum likelihood estimation (MLE):

    P(<next word> | <preceding word sequence of length n>)
    = Count (<preceding word sequence> <next word>)
    / Count (<preceding word sequence>)

  - Example: Count (*They picnicked by the*) / Count (*They picnicked by*)

Bigram probabilities

- For bigrams, the MLE estimate is:

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

  – The count of the occurrences of the sequence $w_{i-1}\ w_i$ divided by the count of the first word w_{i-1}

Example of Bigram probabilities

- Example mini-corpus of three sentences, where we have sentence detection and we include the sentence tags in order to represent the beginning and end of the sentence.

  <S> I am Sam </S>

  <S> Sam I am </S>

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

- Bigram probabilities:

  P (I | <S>) = 2/3 = .67 (probability that I follows <S>)

  P (</S> | Sam) = ½ = .5

  P (Sam | <S>) = 1/3 = .33

  P (Sam | am) = ½ = .5

  P (am | I) = 2/3 = .67

Example using bigram probabilities
to compute the probabilities of sentences:

- Berkeley Restaurant Project sentences
  - 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 Bigram Counts from the corpus

- Out of 9222 sentences, showing counts that the word on the left is followed by the word on the top

	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 probabilities

- Divide/normalize by unigram probabilities:

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

- Result:

	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

Using N-Grams for sentences

- For a bigram grammar $\prod_{k=1}^{n} P(w_k \mid w_{k-1})$

  –P(sentence) can be approximated by multiplying all the bigram probabilities in the sequence

- Example:

P(I want to eat Chinese food) =

P(I | <S>) P(want | I) P(to | want) P(eat | to)
P(Chinese | eat) P(food | Chinese)

More Bigrams from the restaurant corpus

Eat on	.16	Eat Thai	.03
Eat some	.06	Eat breakfast	.03
Eat lunch	.06	Eat in	.02
Eat dinner	.05	Eat Chinese	.02
Eat at	.04	Eat Mexican	.02
Eat a	.04	Eat tomorrow	.01
Eat Indian	.04	Eat dessert	.007
Eat today	.03	Eat British	.001

Examples due to Rada Mihalcea

Additional Bigrams

<S> I	.25	Want some	.04
<S> I'd	.06	Want Thai	.01
<S> Tell	.04	To eat	.26
<S> I'm	.02	To have	.14
I want	.32	To spend	.09
I would	.29	To be	.02
I don't	.08	British food	.60
I have	.04	British restaurant	.15
Want to	.65	British cuisine	.01
Want a	.05	British lunch	.01

Computing Sentence Probabilities

- P(I want to eat British food) = P(I|<S>) P(want|I) P(to|want) P(eat|to) P(British|eat) P(food|British) = .25×.32×.65×.26×.001×.60 = .000080
- vs.
- P(I want to eat Chinese food) = .00015

- Probabilities seem to capture "syntactic" facts, "world knowledge"
  - eat is often followed by a noun
  - British food is not too popular
- N-gram models can be trained by counting and normalization