Natural Language Processing

Jim Martin -- Lecture 3

CSCI 5832

Today

- Language Modeling
  - Probabilistic language models
    - *N*-gram approach
    - Independence assumptions
    - Practical Issues
    - Dealing with zeroes

Word Prediction

- Guess the next word...
  - *So I notice three guys standing on the ___*

What kinds of knowledge did you use to come up with those predictions?

Word Prediction

- We can formalize this task as a problem in discrete probability
  - Given a vocabulary, compute a probability distribution over that vocabulary given the preceding words. $P(w_n|w_{1:n-1})$
  - Or assign a probability to a sequence. $P(w_{1:n})$
  - We'll call a model that can do this a ***Probabilistic Language Model***

Applications

- It turns out that the ability to assess the probability of a sequence is extremely useful. It is at the core of many applications
  - Automatic speech recognition
  - Handwriting and character recognition
  - Spam detection
  - Sentiment analysis
  - Spelling correction
  - Machine translation
  - Summarization

Speech Recognition

- Initial acoustic/signal system proposes two hypotheses for an input sentence
  - *Its hard to wreck a nice beach*
  - *Its hard to recognize speech*
- Job of the language model is to say which of those is more likely

Discrete Probability Review

We're concerned with the *probability of the outcome of discrete events*.

- I flip a coin, what's the probability of it coming up heads?
- What's the probability that it will snow tomorrow?
- What's the probability that school will be closed the following day, given that its snowing when you went to bed?

Discrete Probability Review

- Probabilities are beliefs about an event outcome expressed as a number between 0 and 1.
- The sample space is the set of all possible outcomes.
- An event is some particular outcome.
- A prior is a probability we hold in the absence of any other evidence.
- A conditional is a probability we hold given some set of evidence.

Snow Days

- Probability that school will be closed the following day (C), given that it was snowing when you went to bed (S)
  - P(C | S) “Probability of C given S”

Snow Days

- Probability that school will be closed the following day (C), give that it was snowing when you went to bed (S)
  - $P(C \mid S) = P(C \wedge S) / P(S)$

Snow Days

- Probability that school will be closed the following day (C), give that it was snowing when you went to bed (S)
  - $P(C \mid S) = P(C \wedge S) / P(S)$
  - How would we go about assessing this fraction and what would it mean?

Snow Days

- $P(C \mid S) = P(C \wedge S)/ P(S)$
- Let's look at the parts:
  - $P(C^\wedge S)$ this is a prior probability made up of two events. We want to assess the probability of the intersection of these events. Let's use frequencies. Out of some school records:
    - Count(days that it snowed and school was subsequently closed)/
    - Count(days it snowed in the record)

Snow Days

- $P(C \mid S) = P(C \wedge S)/ P(S)$
- Let's look at the parts:
  - P(S) this is a prior. Let's use frequencies. Out of some school records:
    - Count(days that it snowed)/
    - Count(days in the record)

Snow Days

- $P(C \mid S) = P(C \wedge S) / P(S)$

  $= \text{Count}(\text{closed} \wedge \text{snowed}) / \text{Count}(\text{snowed})$

Out of all the days it snowed, what was the fraction of the days that the schools subsequently closed.

Probability and Language

- With respect to ”language models” we’ll be mainly concerned with the probability of sentences (or sequences of linguistic units)
  - The sentence is the event
  - The sample space is the space of all possible sentences
    - (wait what?)
  - We’d like to assign a probability to that event
    - (this is a strange notion)

Chomsky

"… it must be recognized that the notion of "*probability of a sentence*" is an entirely useless one, under any known interpretation of this term."

"Entirely useless" is a pretty strong claim.
One that turns out to be incorrect.

Language Modeling $P(w_n|w_{1:n-1})$

- How might we go about calculating a conditional probability over word sequences?
  - One way is to use the definition of conditional probabilities and look for counts. So to get
  - P(*the* | *its water is so transparent that*)
- By definition that's

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

- Let's try to get each of those from counts in a large corpus.

Easy Estimate

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

Crude Estimate

- According to Google those counts are 1320 and 1420 so the conditional probability we want is...
  - P(the | its water is so transparent that) = 0.93

Crude Estimate

- How about “matrix”
  - That gives you a 0. 0/1420 = 0
- How about “you”
  - That gives you a 1. 1/1420 = 0.0007
- How about “she”
  - That gives you a 0. 0/1420 = 0
    - This seems wrong. “she” should not be the same as “matrix”

Language Modeling

- Unfortunately, for most sequences, and for most text collections, we won’t get good estimates using counting alone.
  - We’re likely to get a lot of 0 counts, leading to 0 probabilities for sequences that are entirely plausible.
- Clearly, we’ll have to be a more clever to make counting work.
  - First, we’ll use the chain rule for probability
  - And then apply a particularly useful independence assumption

The Chain Rule

- Recall the definition of conditional probabilities

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$

- Rewriting:

$$P(A \wedge B) = P(A \mid B)P(B)$$

- For sequences...
  - $P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$
- In general
  - $P(x_1,x_2,x_3,\ldots x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1\ldots x_{n-1})$

The Chain Rule

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\ldots P(w_n|w_{1:n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

P(its water was so transparent)=

P(its)*

P(water|its)*

P(was|its water)*

P(so|its water was)*

P(transparent|its water was so)

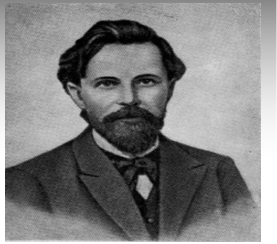
Unfortunately

- There are still a lot of problematically long sequences in this version.
- In general, we'll never be able to get enough data to compute the statistics for those longer prefixes
  - Same problem we had for the original sequence.

Independence Assumption

- Make the simplifying assumption
  - P(lizard|the,other,day,I,was,walking,along,and,saw,a) = P(lizard|a)
- Or maybe
  - P(lizard|the,other,day,I,was,walking,along,and,saw,a) = P(lizard|saw,a)
- That is, the probability in question is to some degree *independent* of its earlier history

Markov Assumption



Replace each component in the product with an approximation (assuming a prefix of size N - 1)

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1})$$

Bigram version

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$$

Estimating Bigram Probabilities

- The Maximum Likelihood Estimate (MLE)

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

Example

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

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

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

$$P(w_n|w_{n-N+1:n-1}) = \frac{C(w_{n-N+1:n-1}\;w_n)}{C(w_{n-N+1:n-1})}$$

Berkeley Restaurant Project

- *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*

Bigram Counts

- Vocabulary size is 1446 |V|
- Out of 9222 sentences
  - "I want" occurred 827 times

	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 bigram counts by the prefix unigram counts to get bigram probabilities.

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

	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

P(want | I) = 827/2533

= .336

Bigram Probabilities

- Divide bigram counts by the prefix unigram counts to get bigram probabilities.

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

	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 english food </s>) =

  P(i|<s>)*

  P(want|I)*

  P(english|want)*

  P(food|english)*

  P(</s>|food)*

  =.000031

Kinds of Knowledge

- As crude as they are, *N*-gram probabilities capture a range of interesting facts about language.

- 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

World knowledge

Syntax

Discourse

Shannon’s Method

- Assigning probabilities to sentences is all well and good, but it’s not terribly illuminating.
- A more entertaining (and very useful) task is to turn the model around and use it to *generate* random sentences that are *similar to* the sentences from which the model was derived.
- Idea enerally attributed to

Claude Shannon.

Shannon's Method (Autoregressive Generation)

- Sample a random bigram (<s>, w_i) according to the models's probability distribution over bigrams
- Now sample a new random bigram (w_i, x) according to its probability. Where the prefix *w matches the suffix of the first bigram chosen.*
- And so on until we randomly choose a (w_i, </s>)
- Then string them together
- <s> I

  I want

  want to

  to eat

  eat Chinese

  Chinese food

  food </s>

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? •Enter Menenius, if it so many good direction found'st thou art a strong upon command of fear not a liberal largess given away, Falstaff! Exeunt
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'd in; • 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 a Corpus

- N=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams...
  - So, 99.96% of the possible bigrams were never seen (have zero entries in the table)
  - This is the biggest problem in language modeling; we'll come back to it.
- Quadrigrams are worse: What's coming out looks like Shakespeare because it *is* Shakespeare

Model Evaluation

- How do we know if our models are any good?
  - And in particular, how do we know if one model is better than another.
- Well Shannon's game gives us an intuition.
  - The generated texts from the higher order models surely sound better.
    - That is, they sound more like the text the model was obtained from.
  - The generated texts from the WSJ and Shakespeare models look very different
    - That is, they look like they' re based on different underlying models.
- But what does that mean? How can we make that notion operational?

Evaluating *N*-Gram Models

- Best evaluation for a language model
  - Put model A into an application
    - For example, a machine translation system
  - Evaluate the performance of the application with model A
  - Put model B into the application and evaluate
  - Compare performance of the application with the two models
    - *Extrinsic evaluation*
    - *A/B Testing*

Evaluation

- Extrinsic evaluation
  - This is quite time consuming and expensive
  - Not something you want to do unless you're pretty sure you've got a good solution
- So
  - As an intermediate evaluation, in order to run rapid experiments, we evaluate N-grams with an *intrinsic* evaluation
  - An evaluation that tries to capture how good the model is intrinsically, not how much it improves performance in some larger system

Evaluation

- Standard method
  - Train parameters of our model on a *training set*.
  - Evaluate the model on some new data: a *test set*.
    - A dataset which is different than our training set, but is drawn from the same source

Perplexity

- Perplexity is just the probability of a test set (assigned by the language model), as normalized by the number of words:

$$\begin{aligned} PP(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N}\,)} \end{aligned}$$

- Chain rule:

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

- For bigrams:

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

- Minimizing model perplexity is the same as maximizing probability of a test set

Perplexity

- The intuition behind perplexity is as a measure is the notion of surprise.
  - How surprised is the language model when it sees the test set?
    - Where surprise is a measure of...
      - Gee, I didn't see that coming...
    - The more surprised the model is, the higher the perplexity
    - The lower the perplexity, the less surprised it was

Lower perplexity is better

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

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Practical Issues

- Once we start looking at test data, we'll run into words that we haven't seen before. So, our models won't work. Standard non-subword solution:
  - Given a corpus
    - Create a fixed lexicon L, of size V
      - Say from a dictionary or
      - A subset of terms from the training set
    - At text normalization phase, any training word not in L is changed to <UNK>
    - Collect counts for that as for any normal word
  - At test time
    - Use UNK counts for any word not seen in training

Practical Issues

- Multiplying a bunch of really small probabilities is a really bad idea.
  - Underflow is likely
- So do everything in log space
  - Avoids underflow (and adding is faster than multiplying)

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