Foundations of Natural Language Processing
Lecture 3
N-gram language models

Alex Lascarides
(Slides based on those from Alex Lascarides and Sharon Goldwater)

21 January 2020

School of informatics

Recap

- Last time, we talked about corpus data and some of the information we can get from it, like word frequencies.
- For some tasks, like sentiment analysis, word frequencies alone can work pretty well (though can certainly be improved on).
- For other tasks, we need more.
- Today: we consider **sentence probabilities**: what are they, why are they useful, and how might we compute them?

Intuitive interpretation

- "Probability of a sentence" = how likely is it to occur in natural language
  - Consider only a specific language (English)
  - Not including meta-language (e.g. linguistic discussion)

$$\text{P(the cat slept peacefully)} > \text{P(slept the peacefully cat)}$$

$$\text{P(she studies morphosyntax)} > \text{P(she studies more faux syntax)}$$

Language models in NLP

- It's very difficult to know the true probability of an arbitrary sequence of words.
- But we can define a **language model** that will give us good approximations.
- Like all models, language models will be good at capturing some things and less good for others.
  - We might want different models for different tasks.
  - Today, one type of language model: an **N-gram model**.

Spelling correction

Sentence probabilities help decide correct spelling.

mis-spelled text		no much effert
↓	(Error model)	
possible outputs		no much effect so much effort no much effort not much effort ...
↓	(Language model)	
best-guess output		not much effort

Automatic speech recognition

Sentence probabilities help decide between similar-sounding options.

speech input		
↓	(Acoustic model)	
possible outputs		She studies morphosyntax She studies more faux syntax She's studies morph or syntax ...
↓	(Language model)	
best-guess output		She studies morphosyntax

Machine translation

Sentence probabilities help decide word choice and word order.

non-English input		
↓	(Translation model)	
possible outputs		She is going home She is going house She is traveling to home To home she is going ...
↓	(Language model)	
best-guess output		She is going home

LMs for prediction

- LMs can be used for **prediction** as well as correction.
- Ex: predictive text correction/completion on your mobile phone.
  - Keyboard is tiny, easy to touch a spot slightly off from the letter you meant.
  - Want to correct such errors as you go, and also provide possible completions. Predict as as you are typing: ineff...
- In this case, LM may be defined over sequences of *characters* instead of (or in addition to) sequences of words.

But how to estimate these probabilities?

- We want to know the probability of word sequence $\vec{w} = w_1 \dots w_n$ occurring in English.
- Assume we have some **training data**: large corpus of general English text.
- We can use this data to **estimate** the probability of $\vec{w}$ (even if we never see it in the corpus!)

Probability theory vs estimation

- Probability theory can solve problems like:
  - I have a jar with 6 blue marbles and 4 red ones.
  - If I choose a marble uniformly at random, what's the probability it's red?

Probability theory vs estimation

- Probability theory can solve problems like:
  - I have a jar with 6 blue marbles and 4 red ones.
  - If I choose a marble uniformly at random, what's the probability it's red?
- But often we don't know the true probabilities, only have data:
  - I have a jar of marbles.
  - I repeatedly choose a marble uniformly at random and then replace it before choosing again.
  - In ten draws, I get 6 blue marbles and 4 red ones.
  - On the next draw, what's the probability I get a red marble?
- First three facts are evidence.
- The question requires estimation theory.

Notation

- I will often omit the random variable in writing probabilities, using $P(x)$ to mean $P(X = x)$.
- When the distinction is important, I will use
  - $P(x)$ for *true* probabilities
  - $\hat{P}(x)$ for *estimated* probabilities
  - $P_{\mathrm{E}}(x)$ for estimated probabilities using a particular estimation method E.
- But since we almost always mean estimated probabilities, I may get lazy later and use $P(x)$ for those too.

Example estimation: M&M colors

What is the proportion of each color of M&M?

- In 48 packages, I find[1] 2620 M&Ms, as follows:

Red	Orange	Yellow	Green	Blue	Brown
372	544	369	483	481	371

- How to estimate probability of each color from this data?

[1] Data from: https://joshmadison.com/2007/12/02/mms-color-distribution-analysis/

Relative frequency estimation

- Intuitive way to estimate discrete probabilities:

$$P_{\mathrm{RF}}(x) = \frac{C(x)}{N}$$

where $C(x)$ is the count of x in a large dataset, and $N = \sum_{x'} C(x')$ is the total number of items in the dataset.

Relative frequency estimation

- Intuitive way to estimate discrete probabilities:

$$P_{\mathrm{RF}}(x) = \frac{C(x)}{N}$$

where $C(x)$ is the count of x in a large dataset, and $N = \sum_{x'} C(x')$ is the total number of items in the dataset.

- M&M example: $P_{\mathrm{RF}}(\text{red}) = \frac{372}{2620} = .142$
- This method is also known as **maximum-likelihood estimation** (MLE) for reasons we'll get back to.

MLE for sentences?

Can we use MLE to estimate the probability of $\vec{w}$ as a sentence of English? That is, the prob that some sentence S has words $\vec{w}$?

$$P_{\mathrm{MLE}}(S = \vec{w}) = \frac{C(\vec{w})}{N}$$

where $C(\vec{w})$ is the count of $\vec{w}$ in a large dataset, and N is the total number of sentences in the dataset.

Sentences that have never occurred

the Archaeopteryx soared jaggedly amidst foliage
vs
jaggedly trees the on flew

- Neither ever occurred in a corpus (until I wrote these slides).
  $\Rightarrow C(\vec{w}) = 0$ in both cases: MLE assigns both zero probability.

- But one is grammatical (and meaningful), the other not.
  $\Rightarrow$ Using MLE on full sentences doesn't work well for language model estimation.

The problem with MLE

- MLE thinks anything that hasn't occurred will never occur (P=0).

- Clearly not true! Such things can have differering, and non-zero, probabilities:
  - My hair turns blue
  - I ski a black run
  - I travel to Finland

- And similarly for word sequences that have never occurred.

Sparse data

- In fact, even things that occur once or twice in our training data are a problem. Remember these words from Europarl?

  cornflakes, mathematicians, pseudo-rapporteur, lobby-ridden, Lycketoft, UNCITRAL, policyfor, Commissioneris, 145.95

  All occurred once. Is it safe to assume all have equal probability?

- This is a **sparse data** problem: not enough observations to estimate probabilities well simply by counting observed data. (Unlike the M&Ms, where we had large counts for all colours!)

- For sentences, many (most!) will occur rarely if ever in our training data. So we need to do something smarter.

Towards better LM probabilities

- One way to try to fix the problem: estimate $P(\vec{w})$ by combining the probabilities of smaller parts of the sentence, which will occur more frequently.

- This is the intuition behind **N-gram language models**.

Deriving an N-gram model

- We want to estimate $P(S = w_1 \dots w_n)$.
  - Ex: $P(S = \text{the cat slept quietly})$.
- This is really a joint probability over the words in S: $P(W_1 = \text{the}, W_2 = \text{cat}, W_3 = \text{slept}, \dots W_4 = \text{quietly})$.
- Concisely, $P(\text{the, cat, slept, quietly})$ or $P(w_1, \dots w_n)$.

Deriving an N-gram model

- We want to estimate $P(S = w_1 \dots w_n)$.
  - Ex: $P(S = \text{the cat slept quietly})$.
- This is really a joint probability over the words in S: $P(W_1 = \text{the}, W_2 = \text{cat}, W_3 = \text{slept}, \dots W_4 = \text{quietly})$.
- Concisely, $P(\text{the, cat, slept, quietly})$ or $P(w_1, \dots w_n)$.
- Recall that for a joint probability, $P(X, Y) = P(Y|X)P(X)$. So,

  $P(\text{the, cat, slept, quietly}) = P(\text{quietly}|\text{the, cat, slept})P(\text{the, cat, slept})$
  $= P(\text{quietly}|\text{the, cat, slept})P(\text{slept}|\text{the, cat})P(\text{the, cat})$
  $= P(\text{quietly}|\text{the, cat, slept})P(\text{slept}|\text{the, cat})P(\text{cat}|\text{the})P(\text{the})$

Deriving an N-gram model

- More generally, the chain rule gives us:

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

- But many of these conditional probs are just as sparse!
  - If we want $P(\text{I spent three years before the mast})$...
  - we still need $P(\text{mast}|\text{I spent three years before the})$.

Example due to Alex Lascarides/Henry Thompson

Deriving an N-gram model

- So we make an **independence assumption**: the probability of a word only depends on a fixed number of previous words (**history**).
  - **trigram model**: $P(w_i|w_1, w_2, \dots w_{i-1}) \approx P(w_i|w_{i-2}, w_{i-1})$
  - **bigram model**: $P(w_i|w_1, w_2, \dots w_{i-1}) \approx P(w_i|w_{i-1})$
  - **unigram model**: $P(w_i|w_1, w_2, \dots w_{i-1}) \approx P(w_i)$
- In our example, a trigram model says
  - $P(\text{mast}|\text{I spent three years before the}) \approx P(\text{mast}|\text{before the})$

Trigram independence assumption

- Put another way, trigram model assumes these are all equal:
  - $P(\text{mast}|\text{I spent three years before the})$
  - $P(\text{mast}|\text{I went home before the})$
  - $P(\text{mast}|\text{I saw the sail before the})$
  - $P(\text{mast}|\text{I revised all week before the})$

  because all are estimated as $P(\text{mast}|\text{before the})$

- Not always a good assumption! But it does reduce the sparse data problem.

Estimating trigram conditional probs

- We still need to estimate $P(\text{mast}|\text{before, the})$: the probability of mast given the two-word history before, the.

- If we use relative frequencies (MLE), we consider:
  - Out of all cases where we saw before, the as the first two words of a trigram,
  - how many had mast as the third word?

Estimating trigram conditional probs

- So, in our example, we'd estimate

$$P_{MLE}(\text{mast}|\text{before, the}) = \frac{C(\text{before, the, mast})}{C(\text{before, the})}$$

  where $C(x)$ is the count of x in our training data.

- More generally, for any trigram we have

$$P_{MLE}(w_i|w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

Example from *Moby Dick* corpus

$$C(\textit{before, the}) = 55 \qquad C(\textit{before, the, mast}) = 4 \qquad \frac{C(\textit{before, the, mast})}{C(\textit{before, the})} = 0.0727$$

- *mast* is the second most common word to come after *before the* in *Moby Dick*; *wind* is the most frequent word.

- $P_{MLE}(\textit{mast})$ is 0.00049, and $P_{MLE}(\textit{mast}|\textit{the})$ is 0.0029.

- So seeing *before the* vastly increases the probability of seeing *mast* next.

Trigram model: summary

- To estimate $P(\vec{w})$, use chain rule and make an indep. assumption:

$$P(w_1, \dots w_n) = \prod_{i=1}^{n} P(w_i|w_1, w_2, \dots w_{i-1})$$
$$\approx P(w_1)P(w_2|w_1)\prod_{i=3}^{n} P(w_i|w_{i-2}, w_{w-1})$$

- Then estimate each trigram prob from data (here, using MLE):

$$P_{MLE}(w_i|w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

- On your own: work out the equations for other N-grams (e.g., bigram, unigram).

Practical details (1)

- Trigram model assumes two word history:

$$P(\vec{w}) = P(w_1)P(w_2|w_1)\prod_{i=3}^{n} P(w_i|w_{i-2}, w_{w-1})$$

- But consider these sentences:

	w_1	w_2	w_3	w_4
(1)	he	saw	the	yellow
(2)	feeds	the	cats	daily

- What's wrong? Does the model capture these problems?

Beginning/end of sequence

- To capture behaviour at beginning/end of sequences, we can augment the input:

	w_{-1}	w_0	w_1	w_2	w_3	w_4	w_5
(1)	<s>	<s>	he	saw	the	yellow	</s>
(2)	<s>	<s>	feeds	the	cats	daily	</s>

- That is, assume $w_{-1} = w_0 =$ <s> and $w_{n+1} =$ </s> so:

$$P(\vec{w}) = \prod_{i=1}^{n+1} P(w_i|w_{i-2}, w_{i-1})$$

- Now, $P($`</s>`$|$`the, yellow`$)$ is low, indicating this is not a good sentence.

Beginning/end of sequence

- Alternatively, we could model all sentences as one (very long) sequence, including punctuation:

two cats live in sam 's barn . sam feeds the cats daily . yesterday , he saw the yellow cat catch a mouse . [...]

- Now, trigrams like $P($`.`$|$`cats daily`$)$ and $P($`,`$|$`. yesterday`$)$ tell us about behavior at sentence edges.

- Here, all tokens are lowercased. What are the pros/cons of *not* doing that?

Practical details (2)

- Word probabilities are typically very small.
- Multiplying lots of small probabilities quickly gets so tiny we can't represent the numbers accurately, even with double precision floating point.
- So in practice, we typically use **negative log probabilities** (sometimes called **costs**):
  - Since probabilities range from 0 to 1, negative log probs range from 0 to ∞: *lower* cost $=$ *higher* probability.
  - Instead of *multiplying* probabilities, we *add* neg log probabilities.

Summary

- "Probability of a sentence": how likely is it to occur in natural language? Useful in many natural language applications.
- We can never know the true probability, but we may be able to estimate it from corpus data.
- N-gram models are one way to do this:
  - To alleviate sparse data, assume word probs depend only on short history.
  - Tradeoff: longer histories may capture more, but are also more sparse.
  - So far, we estimated N-gram probabilites using MLE.

Coming up next

- Weaknesses of MLE and ways to address them (more issues with sparse data).
- How to evaluate a language model: are we estimating sentence probabilities accurately?