Foundations of Natural Language Processing
Lecture 5c
Language Models:
Independence Assumptions

Alex Lascarides

School of Informatics

LMs, MLE and Sparse Data

So far:

- Language Models are very useful
- Want to learn them from data (empirically grounded)
- Sparse Data Problem: 0 probability estimates for possible sequences.
  - the Archaeopteryx soared jaggedly amidst foliage
    vs
    jaggedly trees the on flew

Now: Start to tackle the Sparse Data Problem

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 $P(S = w_1 \ldots w_n)$.
  - Ex: $P(S = \text{the cat slept quietly})$.

- This is really a joint probability over the words in S:
  $P(W_1 = \text{the}, W_2 = \text{cat}, W_3 = \text{slept}, \ldots W_4 = \text{quietly})$.

- Concisely, $P(\text{the, cat, slept, quietly})$ or $P(w_1, \ldots w_n)$.

Deriving an N-gram model

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

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

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

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**: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i|w_{i-2}, w_{i-1})$
  - **bigram model**: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i|w_{i-1})$
  - **unigram model**: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i)$

- 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 $\mathrm{before,\ the}$.

- If we use relative frequencies (MLE), we consider:
  - Out of all cases where we saw $\mathrm{before,\ the}$ 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

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

- *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, $P(\texttt{</s>} | \texttt{the, yellow})$ 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 $P(.|\texttt{cats daily})$ and $P(,|\texttt{. yesterday})$ 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

- Probabilities of word sequences of arbitrary length are 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?