CMPT-413

Computational Linguistics

Anoop Sarkar

http://www.cs.sfu.ca/~anoop

How good is a model

- So far we've seen the probability of a sentence: $P(w_0, \ldots, w_n)$
- What is the probability of a collection of sentences, that is what is the probability of a corpus
- Let $T = s_0, \ldots, s_m$ be a text corpus with sentences s_0 through s_m
- What is $P(T)$?
  Let us assume that we trained $P(\cdot)$ on some *training data*, and T is the *test data*

How good is a model

- $T = s_0, \ldots, s_m$ is the text corpus with sentences s_0 through s_m
- $P(T) = \prod_{i=0}^{m} P(s_i)$
- $P(s_i) = P(w_0^i, \ldots, w_n^i)$
  Let W_T be the length of the text T measured in words
- Cross entropy for T: $H(T) = -\frac{1}{W_T}\log_2 P(T)$
  the average number of bits needed to encode each of the W_T words in the *test data*
  Perplexity: $PP(T) = 2^{H(T)}$

How good is a model

- Lower cross entropy values and perplexity values are better
  Lower values mean that the model is *better*
  Correlation with performance of the language model in various applications

- Performance of a language model is its cross-entropy or perplexity on *test data* (unseen data)
  corresponds to the number bits required to encode that data

- On various real life datasets, typical perplexity values yielded by n-gram models on English text range from about 50 to almost 1000 (corresponding to cross entropies from about 6 to 10 bits/word)

Bigram Models

- In practice:
  $P($Mork read a book$) = P($Mork $\mid \ < start >) \times P($read $\mid$ Mork$) \times$
  $P($a $\mid$ read$) \times P($book $\mid$ a$) \times P(< stop > \ \mid$ book$)$

- $P(w_i \mid w_{i-1}) = \frac{c(w_i, w_{i-1})}{c(w_{i-1})}$
  On unseen data, $c(w_i, w_{i-1})$ or worse $c(w_{i-1})$ could be zero

$$\sum_{w_i} \frac{c(w_i, w_{i-1})}{c(w_{i-1})} = ?$$

Smoothing

- **Smoothing** deals with events that have been observed zero times
- Smoothing algorithms also tend to improve the accuracy of the model

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

- Not just unobserved events: what about events observed once?

Add-one Smoothing

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

- Add-one Smoothing:

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

- Let V be the number of words in our vocabulary
  Remember that we *observe* only V many bigrams
  Assigns count of 1 to unseen bigrams

Add-one Smoothing

$P(\text{Mindy read a book}) = P(\text{Mindy} \mid <start>) \times P(\text{read} \mid \text{Mindy}) \times P(\text{a} \mid \text{read}) \times P(\text{book} \mid \text{a}) \times P(<stop> \mid \text{book})$

- Without smoothing:

$$P(\text{read} \mid \text{Mindy}) = \frac{c(\text{Mindy, read})}{c(\text{Mindy})} = 0$$

- With add-one smoothing (assuming c(Mindy) = 1 but c(Mindy, read) = 0):

$$P(\text{read} \mid \text{Mindy}) = \frac{1}{V + 1}$$

Additive Smoothing: (Lidstone 1920, Jeffreys 1948)

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

- Add-one smoothing works horribly in practice. Seems like 1 is too large a count for unobserved events.

- Additive Smoothing:

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

- $0 < \delta \leq 1$
  Still works horribly in practice, but better than add-one smoothing.

Good-Turing Smoothing: (Good, 1953)

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

- Imagine you're sitting at a sushi bar with a conveyor belt.
- You see going past you 10 plates of tuna, 3 plates of unagi, 2 plates of salmon, 1 plate of shrimp, 1 plate of octopus, and 1 plate of yellowtail
- How likely are you to see a new kind of seafood appear: $\frac{3}{18}$
- How likely are you to see another plate of salmon: should be $< \frac{2}{18}$

Good-Turing Smoothing

- How many types of seafood (words) were seen once? Use this to predict probabilities for unseen events
  Let n_1 be the number of events that occurred once: $p_0 = \frac{n_1}{N}$

- The Good-Turing estimate states that for any n-gram that occurs r times, we should pretend that it occurs r^* times

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

Good-Turing Smoothing

- 10 tuna, 3 unagi, 2 salmon, 1 shrimp, 1 octopus, 1 yellowtail

- How likely is new data? Let n_1 be the number of items occurring once, which is 3 in this case. N is the total, which is 18.

$$p_0 = \frac{n_1}{N} = \frac{3}{18}$$

Good-Turing Smoothing

- 10 tuna, 3 unagi, 2 salmon, 1 shrimp, 1 octopus, 1 yellowtail

- How likely is *octopus*? Since $c(\textit{octopus}) = 1$ The GT estimate is 1^*.

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

- To compute 1^*, we need $n_1 = 3$ and $n_2 = 1$

$$1^* = 2 \times \frac{1}{3} = \frac{2}{3}$$

- What happens when $n_r = 0$?

Simple Backoff Smoothing: incorrect version

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

- In add-one or Good-Turing: $P(\text{the} \mid \text{string}) = P(\text{Fonz} \mid \text{string})$
- If $c(w_i, w_{i-1}) = 0$, then use $P(w_i)$ (back off)
- Works for trigrams: back off to bigrams and then unigrams
- Works better in practice, but probabilities get mixed up (unseen bigrams, for example will get higher probabilities than seen bigrams)

Backoff Smoothing: Jelinek-Mercer Smoothing

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

- $P_{JM}(w_i \mid w_{i-1}) = \lambda P_{ML}(w_i \mid w_{i-1}) + (1 - \lambda) P_{ML}(w_i)$
  where, $0 \leq \lambda \leq 1$

- Notice that $P_{JM}(\text{the} \mid \text{string}) > P_{JM}(\text{Fonz} \mid \text{string})$ as we wanted

- Jelinek-Mercer (1980) describe an elegant form of this **interpolation**:

  $$P_{JM}(n\text{gram}) = \lambda P_{ML}(n\text{gram}) + (1 - \lambda) P_{JM}(n - 1\text{gram})$$

- What about $P_{JM}(w_i)$?

Backoff Smoothing: Many alternatives

$$P_{JM}(n\text{gram}) = \lambda P_{ML}(n\text{gram}) + (1-\lambda)P_{JM}(n-1\text{gram})$$

- Different methods for finding the values for λ correspond to variety of different smoothing methods
  - Katz Backoff (include Good-Turing with Backoff Smoothing)

$$P_{\textit{katz}}(y \mid x) = \begin{cases} \frac{c^*(xy)}{c(x)} & \text{if } c(xy) > 0 \\ \alpha(x)P_{\textit{katz}}(y) & \text{otherwise} \end{cases}$$

  - Deleted Interpolation (Jelinek, Mercer)
    compute λ values from **held-out** data

Backoff Smoothing: Many alternatives

$$P_{JM}(n\text{gram}) = \lambda P_{ML}(n\text{gram}) + (1-\lambda)P_{JM}(n-1\text{gram})$$

- Witten-Bell smoothing
  use the $n-1$gram model when the $n$gram model has too few unique words *in the $n$gram context*

- Absolute discounting (Ney, Essen, Kneser)

$$P_{\text{abs}}(y \mid x) = \begin{cases} \frac{c(xy)-D}{c(x)} & \text{if } c(xy) > 0 \\ \alpha(x)P_{\text{abs}}(y) & \text{otherwise} \end{cases}$$

Backoff Smoothing: Many alternatives

$$P_{JM}(n\text{gram}) = \lambda P_{ML}(n\text{gram}) + (1-\lambda)P_{JM}(n-1\text{gram})$$

- Kneser-Ney smoothing
  $P(\text{Francisco} \mid \text{eggplant}) > P(\text{stew} \mid \text{eggplant})$
  - *Francisco* is common, so interpolation gives $P(\text{Francisco} \mid \text{eggplant})$ a high value
  - But *Francisco* occurs in few contexts (only after *San*)
  - *stew* is common, **and** occurs in many contexts
  - Hence weight the interpolation based on number of contexts for the word using discounting

Backoff Smoothing: Many alternatives

$$P_{JM}(n\text{gram}) = \lambda P_{ML}(n\text{gram}) + (1-\lambda) P_{JM}(n-1\text{gram})$$

- Modified Kneser-Ney smoothing (Chen and Goodman)
  multiple discounts for one count, two counts and three or more counts

- Generalized search (Powell search) or the Expectation-Maximization algorithm

Trigram Models

- Revisiting the trigram model:
  $P(w_1, w_2, \ldots, w_n) =$
  $P(w_1) \times P(w_2 \mid w_1) \times P(w_3 \mid w_1, w_2) \times P(w_4 \mid w_2, w_3) \times$
  $\ldots P(w_i \mid w_{i-2}, w_{i-1}) \ldots \times P(w_n \mid w_{n-2}, \ldots, w_{n-1})$

- Notice that the length of the sentence n is variable

- What is the event space?

The `stop` symbol

- Let $\Sigma = \{a, b\}$ and the language be Σ^*
  so $L = \{\epsilon, a, b, aa, bb, ab, bb \ldots\}$

- Consider a unigram model: $P(a) = P(b) = 0.5$

- $P(a) = 0.5$, $P(b) = 0.5$, $P(aa) = 0.5^2 = 0.25$, $P(bb) = 0.25$
  and so on.

- But $P(a) + P(b) + P(aa) + P(bb) = 1.5$!!

$$\sum_w P(w) = 1$$

The `stop` symbol

- What went wrong?
  No probability for $P(\epsilon)$

- Add a special `stop` symbol:

$$P(a) = P(b) = 0.25$$

$$P(\texttt{stop}) = 0.5$$

- $P(\texttt{stop}) = 0.5$, $P(a\ \texttt{stop}) = P(b\ \texttt{stop}) = 0.25 \times 0.5 = 0.125$, $P(aa\ \texttt{stop}) = 0.25^2 \times 0.5 = 0.03125$ (now the sum is no longer greater than one)

The `stop` symbol

- With this new `stop` symbol we can show that $\sum_w P(w) = 1$
  Notice that the probability of any sequence of length n is $0.25^n \times 0.5$
  Also there are 2^n sequences of length n

$$
\begin{aligned}
& \sum_w P(w) = \\
& \quad \sum_{n=0}^{\infty} 2^n \times 0.25^n \times 0.5 \\
& \quad \sum_{n=0}^{\infty} 0.5^n \times 0.5 = \sum_{n=0}^{\infty} 0.5^{n+1} \\
& \quad \sum_{n=1}^{\infty} 0.5^n = 1
\end{aligned}
$$