Data Intensive Linguistics — Lecture 4
Language Modeling (II): Smoothing and Back-Off

Philipp Koehn

19 January 2006

School of informatics

Language Modeling Example

- Training set

there is a big house
i buy a house
they buy the new house

- Model

$p(big\|a) = 0.5$	$p(is\|there) = 1$	$p(buy\|they) = 1$
$p(house\|a) = 0.5$	$p(buy\|i) = 1$	$p(a\|buy) = 0.5$
$p(new\|the) = 1$	$p(house\|big) = 1$	$p(the\|buy) = 0.5$
$p(a\|is) = 1$	$p(house\|new) = 1$	$p(they\|<s>) = .333$

- Test sentence S: *they buy a big house*

- $p(S) = \underbrace{0.333}_{they} \times \underbrace{1}_{buy} \times \underbrace{0.5}_{a} \times \underbrace{0.5}_{big} \times \underbrace{1}_{house} = 0.0833$

Evaluation of language models

- We want to evaluate the quality of language models
- A good language model gives a high probability to real English
- We measure this with cross entropy and perplexity

Entropy rate of a language

- We want to use entropy and perplexity to measure how well a model explains the test data

- Recall entropy:

$$H(p) = -\sum_x p(x)\ \log p(x)$$

- Entropy over sequences $w_1, ..., w_n$ from a language L:

$$H(w_1, ..., w_n) = -\sum_{W_1^n \in L} p(W_1^n)\ \log p(W_1^n)$$

- Entropy over sequences will depend highly on how long these sequences are. To have a more meaningful measure, we want to measure entropy per word, also called the **entropy rate**:

$$\frac{1}{n}H(w_1, ..., w_n) = -\frac{1}{n}\sum_{W_1^n \in L} p(W_1^n)\ \log p(W_1^n)$$

- To measure **true entropy of a language** L, we need to consider sequences of infinite length

$$\begin{aligned} H(L) &= \lim_{n\to\infty} \frac{1}{n}H(w_1, ..., w_n) \\ &= \lim_{n\to\infty} -\frac{1}{n}\sum_{W_1^n \in L} p(W_1^n)\ \log p(W_1^n) \end{aligned}$$

- This can be simplified (**Shannon-McMillan-Breiman theorem**) to:

$$H(L) = \lim_{n\to\infty} -\frac{1}{n}\log p(W_1^n)$$

- Intuitive explanation: If the sequence is infinite, we do not need to sum over all possible sequences, since the infinite sequence contains all sequences

Cross-entropy

- In practice, we do not have the real probability distribution p for the language L, only a model m for it.

- We define **cross-entropy** (replacing p with m) as

$$H(p, m) = \lim_{n\to\infty} -\frac{1}{n}\log m(W_1^n)$$

- True entropy of a language is an upper bound from cross-entropy:

$$H(p) \leq H(p, m)$$

- Cross entropy is useful measure how well the model fits the true distribution.

Using cross-entropy

- In practice, we do not have an infinite sequence, but a limited test set. However, if the test set is large enough, its *measured* cross-entropy approximates the *true* cross-entropy.

- Example: $p(S) = \underbrace{0.333}_{they} \times \underbrace{1}_{buy} \times \underbrace{0.5}_{a} \times \underbrace{0.5}_{big} \times \underbrace{1}_{house} = 0.0833$

$$\begin{aligned} H(p, m) &= -\frac{1}{5}\log p(S) \\ &= -\frac{1}{5}(\underbrace{\log 0.333}_{they} + \underbrace{\log 1}_{buy} + \underbrace{\log 0.5}_{a} + \underbrace{\log 0.5}_{big} + \underbrace{\log 1}_{house}) \\ &= -\frac{1}{5}(\underbrace{-1.586}_{they} + \underbrace{0}_{buy} + \underbrace{-1}_{a} + \underbrace{-1}_{big} + \underbrace{0}_{house}) = 0.7173 \end{aligned}$$

Perplexity

- **Perplexity** is defined as

$$PP = 2^{H(p,m)}$$
$$= 2^{-\frac{1}{n}\sum_{i=1}^{n} \log m(w_n|w_1,...,w_{n-1})}$$

- In out example $H(m,p) = 0.7173 \Rightarrow PP = 1.6441$
- Intuitively, perplexity is the average number of choices at each point (weighted by the model)
- Perplexity is the most common measure to evaluate language models

Recap from last lecture

- If we estimate probabilities solely from counts, we give probability 0 to unseen events (bigrams, trigrams, etc.)
- One attempt to address this was with add-one smoothing.

Add-one smoothing: results

Church and Gale (1991a) experiment: 22 million words training, 22 million words testing, from same domain (AP news wire), counts of bigrams:

Frequency r in training	Actual frequency in test	Expected frequency in test (add one)
0	0.000027	0.000132
1	0.448	0.000274
2	1.25	0.000411
3	2.24	0.000548
4	3.23	0.000685
5	4.21	0.000822

We overestimate 0-count bigrams $(0.000132 > 0.000027)$, but since there are so many, they use up so much probability mass that hardly any is left.

Using held-out data

- We know from the test data, how much probability mass should be assigned to certain counts.
- We can not use the test data for estimation, because that would be cheating.
- Divide up the training data: one half for count collection, one have for collecting frequencies in unseen text.
- Both halves can be switched and results combined to not lose out on training data.

Deleted estimation

- Counts in training $C_t(w_1, ..., w_n)$
- Counts how often an ngram seen in training is seen in held-out training $C_h(w_1, ..., w_n)$
- Number of ngrams with training count r: N_r
- Total times ngrams of training count r seen in held-out data: T_r
- Held-out estimator:

$$p_h(w_1, ..., w_n) = \frac{T_r}{N_r N} \quad \text{where } count(w_1, ..., w_n) = r$$

Using both halves

- Both halves can be switched and results combined to not lose out on training data

$$p_h(w_1, ..., w_n) = \frac{T_r^{01} + T_r^{10}}{N(N_r^{01} + N_r^{10})} \quad \text{where } count(w_1, ..., w_n) = r$$

Deleted estimation: results

- Much better:

Frequency r in training	Actual frequency in test	Expected frequency in test (Good Turing)
0	0.000027	0.000037
1	0.448	0.396
2	1.25	1.24
3	2.24	2.23
4	3.23	3.22
5	4.21	4.22

- Still overestimates unseen bigrams (why?)

Good-Turing discounting

- Method based on the assumption of binomial distribution of frequencies.
- Translate real counts r for words with adjusted counts r^*:

$$r^* = (r+1)\frac{E(N_{r+1})}{E(N_r)}$$

N_r is the *count of counts*: number of words with frequency r.

- The probability mass reserved for unseen events is $E(N_1)/N$.
- For large r (where N_{r-1} is often 0), so various other methods can be applied (don't adjust counts, curve fitting to linear regression). See Manning+Schütze for details.

Good-Turing discounting: results

- Almost perfect:

Frequency r in training	Actual frequency in test	Expected frequency in test (Good Turing)
0	0.000027	0.000027
1	0.448	0.446
2	1.25	1.26
3	2.24	2.24
4	3.23	3.24
5	4.21	4.22

Is smoothing enough?

- If two events (bigrams, trigrams) are both seen with the same frequency, they are given the same probability.

n-gram	count
scottish beer is	0
scottish beer green	0
beer is	45
beer green	0

- If there is not sufficient evidence, we may want to **back off** to lower-order n-grams

Combining estimators

- We would like to use high-order n-gram language models
- ... but there are many ngrams with count 0.

→ Linear interpolation p_{li} of estimators p_n of different order n:

$$\begin{aligned} p_{li}(w_n|w_{n-2}, w_{n-1}) = \; & \lambda_1 \; p_1(w_n) \\ & + \lambda_2 \; p_2(w_n|w_{n-1}) \\ & + \lambda_3 \; p_1(w_n|w_{n-2}, w_{n-1}) \end{aligned}$$

- $\lambda_1 + \lambda_2 + \lambda_3 = 1$

Katz's backing-off

- Another approach is to back-off to lower order n-gram language models

$$p_{bo}(w_n|w_{n-2}, w_{n-1}) = \begin{cases} (1 - d(w_{n-2}, w_{n-1})) \; p(w_{n-2}, w_{n-1}) \\ \quad \text{if } count(w_{n-2}, w_{n-1}) > 0 \\ \alpha(w_{n-2}, w_{n-1}) \; p_{bo}(w_n|w_{n-1}) \\ \quad \text{otherwise} \end{cases}$$

- The weight $\alpha(w_{n-2}, w_{n-1})$ given to the back-off path has to be chosen appropriately. Because this gives probability mass to unseen events, the maximum likelihood estimate has to be discounted (by $d(w_{n-2}, w_{n-1})$)

General linear interpolation

- We can generalize interpolation and back-off:

$$\begin{aligned} p_{li}(w_n|w_{n-2}, w_{n-1}) = \; & \lambda_1(w_{n-2}, w_{n-1}) \; p_1(w_n) \\ & + \lambda_2(w_{n-2}, w_{n-1}) \; p_2(w_n|w_{n-1}) \\ & + \lambda_3(w_{n-2}, w_{n-1}) \; p_1(w_n|w_{n-2}, w_{n-1}) \end{aligned}$$

- How do we set the λs ?

Consideration for weights $\lambda(w_{n-2}, w_{n-1})$

- Based on $count(w_{n-2}, w_{n-1})$: the more frequent the history, the higher λ.

→ Organize histories in bins with similar counts, and optimize the resulting few $\lambda(bin(w_{n-2}, w_{n-1}))$ by optimizing perplexity on a limited **development set**

- Also consider entropy of predictions:
  - both *great deal* and *of that* occur 178 times in a selection of novels by Jane Austin
  - *of that* is followed by 115 different words
  - *great deal* is followed by 36 different words, 38% of the time *of* follows

Other methods in language modeling

- Language modeling is still an active field of research
- There are many back-off and interpolation methods
- Skip n-gram models: back-off to $p(w_n|w_{n-2})$
- Factored language models: back-off to word stems, part-of-speech tags
- Syntactic language models: using parse trees
- Language models trained on 200 billion words using 2 TB disk space