Foundations of Natural Language Processing
Lecture 7a: More smoothing

Alex Lascarides

Recap: Smoothing for language models

- N-gram LMs reduce sparsity by assuming each word only depends on a fixed-length history.

- But even this assumption isn't enough: we still encounter lots of unseen N-grams in a test set or new corpus.

- If we use MLE, we'll assign 0 probability to unseen items: useless as an LM.

- **Smoothing** solves this problem: move probability mass from seen items to unseen items.

Smoothing methods so far

- Add-α smoothing: ($\alpha = 1$ or < 1) very simple, but no good when vocabulary size is large.

- Good-Turing smoothing:
  - estimate the probability of seeing (any) item with N_c counts (e.g., 0 count) as the proportion of items already seen with N_{c+1} counts (e.g., 1 count).
  - Divide that probability evenly between all possible items with N_c counts.

Good-Turing smoothing

- If n is count of history, then $P_{GT} = \frac{c^*}{n}$ where

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

  - N_c number of N-grams that occur exactly c times in corpus
  - N_0 total number of unseen N-grams

- Ex. for trigram probability $P_{GT}(\text{three}|\text{I spent})$, then n is count of $\mathrm{I\ spent}$ and c is count of $\mathrm{I\ spent\ three}$.

Problems with Good-Turing

- Assumes we know the vocabulary size (no unseen words)
  [but again, use UNK: see J&M 4.3.2]

- Doesn't allow "holes" in the counts (if $N_i > 0$, $N_{i-1} > 0$)
  [can estimate using linear regression: see J&M 4.5.3]

- Applies discounts even to high-frequency items
  [but see J&M 4.5.3]

- But there's a more fundamental problem...

Remaining problem

- In training corpus, suppose we see Scottish beer but neither of
  - Scottish beer drinkers
  - Scottish beer eaters
- If we build a trigram model smoothed with Add-α or G-T, which example has higher probability?

Remaining problem

- Previous smoothing methods assign equal probability to all unseen events.
- Better: use information from lower order N-grams (shorter histories).
  - beer drinkers
  - beer eaters
- Two ways: **interpolation** and **backoff**.

Interpolation

- **Combine** higher and lower order N-gram models, since they have different strengths and weaknesses:
  - high-order N-grams are sensitive to more context, but have sparse counts
  - low-order N-grams have limited context, but robust counts
- If P_N is N-gram estimate (from MLE, GT, etc; $N = 1 - 3$), use:

$$P_{\text{INT}}(w_3|w_1, w_2) = \lambda_1 \; P_1(w_3) + \lambda_2 \; P_2(w_3|w_2) + \lambda_3 \; P_3(w_3|w_1, w_2)$$

$$P_{\text{INT}}(\text{three}|\text{I, spent}) = \lambda_1 \; P_1(\text{three}) + \lambda_2 \; P_2(\text{three}|\text{spent}) + \lambda_3 \; P_3(\text{three}|\text{I, spent})$$

Interpolation

- Note that λ_is must sum to 1:

$$
\begin{aligned}
1 &= \sum_{w_3} P_{\mathrm{INT}}(w_3|w_1, w_2) \\
&= \sum_{w_3} [\lambda_1\ P_1(w_3) + \lambda_2\ P_2(w_3|w_2) + \lambda_3\ P_3(w_3|w_1, w_2)] \\
&= \lambda_1 \sum_{w_3} P_1(w_3) + \lambda_2 \sum_{w_3} P_2(w_3|w_2) + \lambda_3 \sum_{w_3} P_3(w_3|w_1, w_2) \\
&= \lambda_1 + \lambda_2 + \lambda_3
\end{aligned}
$$

Fitting the interpolation parameters

- In general, any weighted combination of distributions is called a **mixture model**.

- So λ_is are **interpolation parameters** or **mixture weights**.

- The values of the λ_is are chosen to optimize perplexity on a held-out data set.

Katz Back-Off

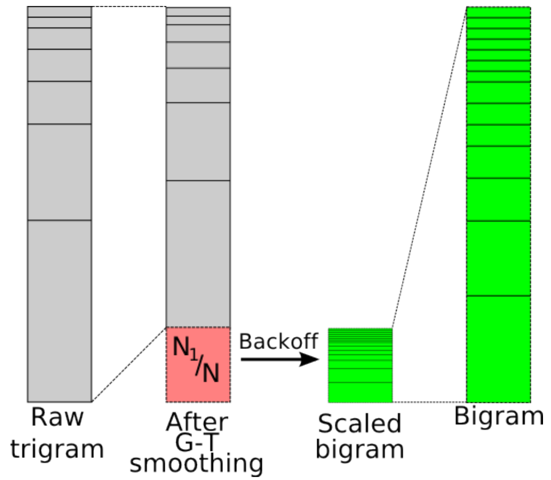
- Solve the problem in a similar way to Good-Turing smoothing.

- Discount the trigram-based probability estimates.

- This leaves some probability mass to share among the estimates from the lower-order model(s).

- Katz backoff: Good-Turing discount the observed counts, but

- instead of distributing that mass uniformly over unseen items, use it for backoff estimates.



Back-Off Formulae

- Trust the highest order language model that contains N-gram

$$P_{BO}(w_i|w_{i-N+1},...,w_{i-1}) =$$

$$= \begin{cases} P^*(w_i|w_{i-N+1},...,w_{i-1}) \\ \qquad\qquad\qquad \text{if count}(w_{i-N+1},...,w_i) > 0 \\ \alpha(w_{i-N+1},...,w_{i-1})\ P_{BO}(w_i|w_{i-N+2},...,w_{i-1}) \\ \qquad\qquad\qquad \text{else} \end{cases}$$

Back-Off

- Requires
  - adjusted prediction model $P^*(w_i|w_{i-N+1}, ..., w_{i-1})$
  - backoff weights $\alpha(w_1, ..., w_{N-1})$
- Exact equations get complicated to make probabilities sum to 1.
- See textbook for details if interested.

Summary

- Laplace and Good Turing recognise that a non-zero prob. mass is requires on unseen ngrams.

- Interpolation/backoff: leverage advantages of both higher and lower order N-grams.