Natural Language Processing

# Smoothing and perplexity

Alla Rozovskaya

Slides are based on material in J&M and Michael Collins' NLP course at Columbia.

# Previous lecture

- Language models
  - Word prediction
  - Key probability terminology
    - Joint probability
    - Chain rule/conditional probability
    - Independence assumptions (Markov processes)
  - N-gram models: unigrams, bigrams, trigrams
  - Parameter estimation from a training corpus
  - Maximum likelihood estimate (relative frequencies)
  - Sparse data problems

# Evaluating language models

- Parameters of a language model are estimated on a training corpus
- Then we can evaluate how well the LM fits our (test) data that contains m sentences:

$$\prod_{i=1}^{m} p(s_i)$$

# Perplexity

- Instead of the product of probabilities, we will have the log:

$$\log \prod_{i=1}^{m} p(s_i) = \sum_{i=1}^{m} \log p(s_i)$$

- Then the quality of the LM is estimated using the notion of **perplexity** (related to entropy)

$$Perplexity = 2^{-l}$$

# Perplexity

- First, we compute the **average log probabilities word-by-word** in the test data (M is the number of words (tokens) in the test data):

$$l = \frac{1}{M} \sum_{i=1}^{m} \log p(s_i)$$

# Perplexity

- Lower perplexity corresponds to a better fit of the language model

# Perplexities and N-gram order

- Unigram language models will have the poorest fit since they generate every word independently
- Bigram language model will have a better fit as they condition on the previous word
- Trigram LMs will fit better given a sufficient amount of training data to estimate model parameters
- Trigrams are hard to beat with more sophisticated models that condition on more events

# Sparse data problems

- When training even on a very large corpus, a lot of N-grams will not occur and thus will have 0 probability under the MLE estimate:
  - Bigram $p(w_i | w_{i-1})=c(w_{i-1}, w_i) / c(w_{i-1})$
  - Trigram $p(w_i | w_{i-2}, w_{i-1})$

    $= c(w_{i-2}, w_{i-1}, w_i)/c(w_{i-2}, w_{i-1})$
- Such models will not be able to fit the test data well.

# Smoothing

- Smoothing is a technique that allows us to reserve some probability mass and distribute it to “unseen” events
  - Interpolation
  - Discounting

Basic idea: “discount” probability mass from seen events and distribute to unseen events.

# Add-One smoothing

- Add 1 to all counts
- Example (unigrams):
  - MLE estimate (**N** is the number of tokens in the training data):
    - $p(w_i)=c(w_i)/N$
  - $c^*(w_i)=c(w_i)+1$
  - Then $p^*(w_i)=(c(w_i)+1)/(N+V)$ where **V** is the number of word types (vocabulary size)

# Add-One smoothing (bigrams)

- Recall: MLE bigram probabilities are computed as follows:

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

- Then with the Add-One smoothing:

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

# Add-One smoothing

- **General idea**: by adding a count of 1 to every word type including those with 0 counts, we are taking away some probability mass from observed events and re-assigning it to *unseen* events

# Add-One smoothing

- Problem: too much probability mass is assigned to unseen events
- This is a poor method of smoothing

# Linear interpolation

- In a trigram model, we need to estimate:

$$p(w_i \mid w_{i-2}, w_{i-1})$$

- MLE gives us:

$$p(w_i \mid w_{i-2}, w_{i-1}) = \frac{c(w_{i-2}, w_{i-1}, w_i)}{c(w_{i-2}, w_{i-1})}$$

- What happens when either of the counts did not occur?

# Linear interpolation

- Basic idea: instead of just using trigram counts, interpolate these with lower-order counts:

$$p(w_i \mid w_{i-2}, w_{i-1}) = \lambda_1 \cdot p_{ML}(w_i \mid w_{i-2}, w_{i-1}) + \lambda_2 \cdot p_{ML}(w_i \mid w_{i-1}) + \lambda_3 \cdot p_{ML}(w_i)$$

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

  $\lambda_i \geq 0$

# Linear interpolation

- Values of $\lambda_i$ can be learned on a separate development (tuning) set of data

# Discounting methods

| *x* | Count(x) | ML bigram estimate p(w\|"the") |
|---|---|---|
| the | 48 | |
| | | |
| the,dog | 15 | 15/48 |
| the,woman | 11 | 11/48 |
| the,man | 10 | 10/48 |
| the,park | 5 | 5/48 |
| the,job | 2 | 2/48 |
| the,telescope | 1 | 1/48 |
| the,manual | 1 | 1/48 |
| the,afternoon | 1 | 1/48 |
| the,country | 1 | 1/48 |
| the,street | 1 | 1/48 |

These ML estimates are going to be high (especially for low-frequency events).

# Discounted counts

- Count*(x)=Count(x)-0.5
- New estimates

| *x* | **Count(x)** | **Count*(x)** | **New discounted bigram estimate** $\frac{Count^*(x)}{Count("the")}$ |
|---|---|---|---|
| the | 48 | 48 | |
| | | | |
| the,dog | 15 | 14.5 | 14.5/48 |
| the,woman | 11 | 10.5 | 10.5/48 |
| the,man | 10 | 9.5 | 9.5/48 |
| the,park | 5 | 4.5 | 4.5/48 |
| the,job | 2 | 1.5 | 1.5/48 |
| the,telescope | 1 | 0.5 | 0.5/48 |
| the,manual | 1 | 0.5 | 0.5/48 |
| the,afternoon | 1 | 0.5 | 0.5/48 |
| the,country | 1 | 0.5 | 0.5/48 |
| the,street | 1 | 0.5 | 0.5/48 |

# Discounted estimates

- The discounting method lowers the estimates of the observed bigrams
- But now we have some leftover probability mass: $\frac{43}{48} < 1$

| *x* | Count(x) | Count*(x) | New discounted bigram estimate $\frac{Count*(x)}{Count(\text{"the"})}$ |
|---|---|---|---|
| the | 48 | 48 | |
| | | | |
| the,dog | 15 | 14.5 | 14.5/48 |
| the,woman | 11 | 10.5 | 10.5/48 |
| the,man | 10 | 9.5 | 9.5/48 |
| the,park | 5 | 4.5 | 4.5/48 |
| the,job | 2 | 1.5 | 1.5/48 |
| the,telescope | 1 | 0.5 | 0.5/48 |
| the,manual | 1 | 0.5 | 0.5/48 |
| the,afternoon | 1 | 0.5 | 0.5/48 |
| the,country | 1 | 0.5 | 0.5/48 |
| the,street | 1 | 0.5 | 0.5/48 |

# Remaining probability mass

$$1 - \frac{43}{48} = \frac{5}{48}$$

We can now take this **leftover probability** and divide it among the words that were never seen following "the" in our training corpus.

$$\alpha(w_{i-1}) = 1 - \sum_{w} \frac{Count^*(w_{i-1}, w_i)}{Count(w_{i-1})}$$

**Leftover probability**

$$\alpha(\text{"}the\text{"}) = \frac{5}{48}$$

# Interpolation vs. backoff

- **Interpolation**: combine estimates from higher and lower-order counts
- **Backoff**: use discounting to “backoff” to lower-order n-grams when there is no evidence for higher-order n-grams in the training data:
  - Estimate trigram probabilities by using bigram probabilities
  - Estimate bigrams by using unigram probabilities

# Katz backoff

- If we have non-zero counts, we rely on these
- Otherwise, we “back off” to a lower-order count (and do not interpolate these)

# Katz backoff models (bigrams)

- For word $w_{i-1}$, define two sets of words:

$$A(w_{i-1}) = \{w : Count(w_{i-1}, w_i) > 0\}$$
$$B(w_{i-1}) = \{w : Count(w_{i-1}, w_i) = 0\}$$

Then backed-off bigram estimates are computed as follows:

$$p_{BO}(w_i \mid w_{i-1}) = \begin{cases} \dfrac{Count^*(w_{i-1}, w_i)}{Count(w_{i-1})}, w_i \in A(w_{i-1}) \\ \alpha(w_{i-1}) \dfrac{p_{ML}(w_i)}{\sum_{w \in B(w_{i-1})} p_{ML}(w)}, w_i \in B(w_{i-1}) \end{cases}$$

**Leftover probability is distributed proportionally to unigram counts**

# Katz backoff models (trigrams)

- For bigram ($w_{i-2}$, $w_{i-1}$), define two sets of words:

$$A(w_{i-2}, w_{i-1}) = \{w : Count(w_{i=2}, w_{i-1}, w_i) > 0\}$$

$$B(w_{i-2}, w_{i-1}) = \{w : Count(w_{i=2}, w_{i-1}, w) = 0\}$$

$Count*(w_{i-1}, w_i$

Then backed-off trigram estimates are computed as follows:

$$p_{BO}(w_i \mid w_{i-2}, w_{i-1}) = \begin{cases} \dfrac{Count*(w_{i-2}, w_{i-1}, w_i)}{Count(w_{i-2}, w_{i-1})}, w_i \in A(w_{i-2}, w_{i-1}) \\ \alpha(w_{i-2}, w_{i-1}) \dfrac{p_{ML}(w_i \mid w_{i-1})}{\sum_{w \in B(w_{i-2}, w_{i-1})} p_{ML}(w \mid w_{i-1})}, w_i \in B(w_{i-2}, w_{i-1}) \end{cases}$$

**Leftover probability is distributed proportionally to bigram counts**

# Discounted probability (trigrams)

- Bigrams: $\alpha(w_{i-1}) = 1 - \sum_{w \in A(w_{i-1})} \frac{Count^*(w_{i-1}, w_i)}{Count(w_{i-1})}$

- Trigrams: $\alpha(w_{i-2}, w_{i-1}) = 1 - \sum_{w \in A(w_{i-2}, w_{i-1})} \frac{Count^*(w_{i-2}, w_{i-1}, w_i)}{Count(w_{i-2}, w_{i-1})}$

# Discounting parameter

- Can be obtained empirically on a tuning set
- Higher counts are said to be more reliable than low counts (e.g. <5), so it’s possible to only discount these unreliable counts.