Assignment 5 + Smoothing 2
(SNLP Tutorial 6)

Vilém Zouhar, Awantee Deshpande, Julius Steuer

1st, 2nd June 2021

Assignment 5

- Exercise 1: OOV Words
- Exercise 2: Additive smoothing
- Exercise 3: Perplexity, infinite smoothing, interpolation
- Bonus: Other language models

Cross-validation

- K-fold cross-validation: Divide data into k subsets, train on k-1 subsets and test on the remaining 1.
- Leave One Out cross-validation: Train on all data points except one. Do this N times.

Questions

- Why is cross-validation beneficial?
- How does shuffling the dataset affect the LOOV score?
- When is k-fold cross-validation beneficial over standard cross-validation?

Smoothing Techniques

Remember the basics!

We perform smoothing to keep a language model from assigning 0 or $\sim$0 probabilities to rare/unseen events.

Different ways to do this...

Floor Discounting

$$P(w|h) = \frac{N(w,h) + \epsilon}{N(h) + \epsilon \cdot V}$$

Variants: Laplace smoothing, Lidstone smoothing, add-α smoothing...

Good-Turing

Data: 🍏 🍏 🍏 🍆 🍏 🍌 🍌 🍒 🍏 🍆 🍌 🍌 🍒 🍆 🍇 🌿

- $N_4 = \{$🍌$\}$
- $N_3 = \{$🍏, 🍆$\}$
- $N_2 = \{$🍒$\}$
- $N_1 = \{$🍇, 🌿$\}$
- $N_0 = \{$🍨$\}$

$$p_r = \frac{(r+1)N_{r+1}}{N_r} \cdot \frac{1}{N}$$

- Nominator: expected total number of occurences of words that occur $r+1$ times
- Denominator-left: previous bucket size
- Fraction-left: expected number of occurences of a single word from that bucket
- Denominator-right: divide by total occurences

Good-Turing - Questions

- Let k be the maximum occurence of a word. What's the issue?
- A similar issue related to the one above?
- Do the probabilities sum up to 1?
- How to make it work for anything above unigrams?

Linear Intepolation/Jelinek-Mercer smoothing

B_1: (FROZEN YOGHURT)

B_2: (FROZEN RED)

What will floor discounting do here? Can we interpolate our bigram model with a unigram model?

$$P(w|h) = \lambda_1 P(w|h) + (1 - \lambda_1)P(w)$$

Can be generalised to higher order n-grams.

Questions

- What condition must be fulfilled for higher n-grams?
- How is λ_i determined?
- Can you smooth the above probabilities?

Backing-Off models

What other way can we use the lower-order n-gram distributions? Is a lot of context always a good thing?

Idea behind back-off models: Use information from a lower order n-gram distribution.

A "recursion" strategy...

$$P(w|h) = \begin{cases} \frac{N(w,h)-d}{N(h)} + \alpha(h)\beta(w|h) & \text{for N(w,h)} > 0 \\ \alpha(h)\beta(w|h) & \text{otherwise} \end{cases} \tag{1}$$

Absolute Discounting

Corpus

- Train set:
  🍏 🍏 🍏 🍆 🍏 🍌 🍌 🍒 🍏 🍆 🍌 🍌 🍒 🍌 🍏 🍆
- Test set:
  🍫 🍏 🍟 🍌 🍏 🍆 🍆 🍌 🍒 🍟 🍏 🍏

Distribution

- Vocabulary counts

🍏 6 🍌 5 🍆 3 🍒 2 🍟 0 🍫 0

- Decrease all non-zero counts by some parameter d = 0.75

🍏 6−0.75 🍌 5−0.75 🍆 3−0.75 🍒 2−0.75 🍟 0 🍫 0

- Divide by $N = 16$

🍏 0.33 🍌 0.26 🍆 0.14 🍒 0.11 🍟 0 🍫 0

Sum $= 0.33+0.26+0.14+0.11 = 0.84 \neq 1$.

Idea: Utilise this probability mass for zero counts.

Absolute Discounting

$$P(w|h) = \frac{c(w,h) - d}{c(h)}$$

Adjust the probability mass $1 - \sum_h \frac{c(w,h)-d}{c(h)}$

e.g. For bigrams,

$$P_{abs}(w_i|w_{i-1}) = \frac{max N(w_{i-1}, w_i) - d, 0}{\sum_{w'} N(w_{i-1}, w')} + \lambda(w_{i-1}) P_{abs}(w_i)$$

$$P_{abs}(w_i) = \frac{max N(w_i) - d, 0}{\sum_{w'} N(w')} + \lambda(.) P_{unif}(w_i)$$

$$\texttt{where } \lambda(w_{i-1}) = \frac{d}{\sum_{w'} N(w_{i-1}, w')} \cdot N_{1+}(w_{i-1}, \bullet)$$

$$\lambda(.) = \frac{d}{\sum_{w'} N(w')} \cdot N_{1+}$$

Kneser-Ney Smoothing

TODO

Pruning

TODO

Assignment 6

- Exercise 1: MAP and MLE estimates
- Exercise 2: Good Turing Smoothing
- Exercise 3: Cross-Validation

Resources

1. UdS SNLP Class: https://teaching.lsv.uni-saarland.de/snlp/
2. n-gram models: https://web.stanford.edu/~jurafsky/slp3/3.pdf
3. Twitter emojis