Homework 1 Problem Set Solutions

Ankur Parikh

October 22, 2018

- Characterize the inference time complexity of Kneser Ney language models in big-O notation. You should decide what variables are important to model when computing these two aspects.

  Let us consider the bigram case for simplicity since it is easy to see how it extrapolates.

  $$P_{kn}(w|w_{i-1}) = \frac{\max(c(w_{i-1}, w) - d, 0)}{\sum_w c(w_{i-1}, w)} + \alpha(w_{i-1}) \frac{|\{w : c(w_{i-1}, w) > 0\}|}{\sum_w |\{w : c(w_{i-1}, w) > 0\}|}$$

  where $\alpha(w_{i-1}) = \sum_w \frac{d \times |\{w : c(w_{i-1}, w) > 0\}|}{\sum_w c(w_{i-1}, w)}$ is the leftover probability from discounting.

  Consider the following three terms:

  1. $\frac{\max(c(w_{i-1}, w) - d, 0)}{\sum_w c(w_{i-1}, w)}$
  2. $\alpha(w_{i-1})$
  3. $\frac{|\{w : c(w_{i-1}, w) > 0\}|}{\sum_w |\{w : c(w_{i-1}, w) > 0\}|}$

  Note that each can be precomputed from the training data and stored in a hash map (or properly indexed matrix etc.) for constant time lookup. Thus for each n-gram the inference complexity is $O(1)$ and if we have an n-gram model then the inference complexity will be $O(n)$.

  We cannot efficiently store $P_{kn}(w|w_{i-1})$ directly for $O(1)$ lookup since it is dense and would take too much memory, whereas the 3 individual components described above are sparse.

- Characterize the memory complexity of Kneser Ney language models in big-O notation. You should decide what variables are important to model when computing these two aspects.

  The brute force complexity (and incorrect answer) is $O(V^n)$ where V is the vocabulary size if you store the counts densely. This answer is incorrect because it is very inefficient since the vocabulary is typically around a million.

To be more efficient, realize that we basically need to store modified versions of counts e.g. $c(w_{i-1}, w)$ or $\sum_w c(w_{i-1}, w)$. Thus the memory complexity is contingent on the number of non-zero elements in these count matrices/tensors which is equal to the number of unique n-grams for each n. Note that the number of unique n-grams for any n is upper bounded by the number of tokens in the dataset (i.e. if we have 1 million tokens, then we will have at most 1 million unique bigrams for instance).

This gives a complexity of $O(n|D|)$ where $|D|$ is the number of tokens in the training set (which is much smaller than $O(V^n)$ under reasonable vocabulary / dataset size).