COMS W4705, Fall 2011: Problem Set 1
Analytical Solutions

- Total points: 40
- Due date: Friday, September 30th at 5pm
- Late policy: 5 points off for every day late, 0 points if handed in after 5pm on October 3rd.

Question 1 (15 points)

In lecture we saw a method for language modeling called *linear interpolation*, where the trigram estimate $q(w_i \mid w_{i-2}, w_{i-1})$ is defined as

$$q(w_i \mid w_{i-2}, w_{i-1}) = \lambda_1 \times q_{ML}(w_i \mid w_{i-2}, w_{i-1}) + \lambda_2 \times q_{ML}(w_i \mid w_{i-1}) + \lambda_3 \times q_{ML}(w_i)$$

Here $\lambda_1, \lambda_2, \lambda_3$ are weights for the trigram, bigram, and unigram estimates, and q_{ML} stands for the maximum-likelihood estimate.

One way to optimize the λ values (again, as seen in lecture), is to use a set of validation data, in the following way. Say the validation data consists of n sentences, $S_1, S_2, \ldots, S_n$. Define $\mathrm{c}'(w_1, w_2, w_3)$ to be the number of times the trigram w_1, w_2, w_3 is seen in the validation sentences. Then the λ values are chosen to maximize the following function:

$$L(\lambda_1, \lambda_2, \lambda_3) = \sum_{w_1, w_2, w_3} \mathrm{c}'_2(w_1, w_2, w_3) \log q(w_3, |w_1, w_2)$$

Question: show that choosing λ values that maximize $L(\lambda_1, \lambda_2, \lambda_3)$ is equivalent to choosing λ values that *minimize the perplexity* of the language model on the validation data.

Solution

Definitions:

$S_i = w_{i,1}, w_{i,2} \ldots w_{i,n_i}$

m is the number of sentences and n_i is the length of the ith sentence, $w_{i,j}$ is the jth word in the ith sentence. w_1, w_2, w_3 are examples of words in the ith sentence.

M is the total number of words in the corpus, or $\sum_i n_i$.

In class we defined perplexity as 2^{-l} where $l = \frac{1}{M} \sum_i \log p(S_i)$. This means that we want to minimize 2^{-l}, or maximize $l = \frac{1}{M} \sum_i \log p(S_i)$.

$$\begin{aligned} p(S_i) &= \prod_{j=1}^{n_i} q(w_{i,j} | w_{i,j-2}, w_{i,j-1}) \\ \log p(S_i) &= \sum_{j=1}^{n_i} \log q(w_{i,j} | w_{i,j-2}, w_{i,j-1}) \end{aligned}$$

$$\sum_i \log p(S_i) = \sum_{i=1}^{m} \sum_{j=1}^{n_i} \log q(w_{i,j} | w_{i,j-2}, w_{i,j-1})$$

The above equation is for all the words in all the sentences, treating each word as the last word in the trigram. This can be rewritten to look at each trigram, (w_1, w_2, w_3) as follows:

$$\begin{aligned} \sum_i \log p(S_i) &= \sum_{w_1,w_2,w_3} c'(w_1, w_2, w_3) \log q(w_3 | w_1, w_2) \\ \sum_i \log p(S_i) &= L(\lambda_1, \lambda_2, \lambda_3) \end{aligned}$$

As mentioned earlier, we want to maximize $l = \frac{1}{M} \sum_i \log p(S_i)$ which we have just shown is equivalent to maximizing $L(\lambda_1, \lambda_2, \lambda_3)$. (Note: $\frac{1}{M}$ can be disregarded because it is a constant value - the number of words in the corpus).

Question 2 (10 points)

In the lecture we saw an improved method for linear interpolation where the λ values are sensitive to the number of times the bigram (w_{i-2}, w_{i-1}) has been seen; the intuition behind this was that the more frequently this bigram has been seen, the more weight should be put on the trigram estimate.

Here we'll define a method that is similar in form to the method seen in lecture, but differs in some important ways. First, we define a function $\Phi(w_{i-2}, w_{i-1}, w_i)$ which maps trigrams into "bins", depending on their count. For example, we can define Φ as follows:

$$\begin{aligned} &\Phi(w_{i-2}, w_{i-1}, w_i) = 1 \quad \text{If} \quad Count(w_{i-2}, w_{i-1}, w_i) = 0 \\ &\Phi(w_{i-2}, w_{i-1}, w_i) = 2 \quad \text{If} \quad 1 \leq Count(w_{i-2}, w_{i-1}, w_i) \leq 2 \\ &\Phi(w_{i-2}, w_{i-1}, w_i) = 3 \quad \text{If} \quad 3 \leq Count(w_{i-2}, w_{i-1}, w_i) \leq 5 \\ &\Phi(w_{i-2}, w_{i-1}, w_i) = 4 \quad \text{If} \quad 6 \leq Count(w_{i-2}, w_{i-1}, w_i) \end{aligned}$$

The trigram estimate $q(w_i \mid w_{i-2}, w_{i-1})$ is then defined as

$$\begin{aligned} q(w_i \mid w_{i-2}, w_{i-1}) = &\lambda_1^{\Phi(w_{i-2},w_{i-1},w_i)} \times q_{ML}(w_i \mid w_{i-2}, w_{i-1}) \\ &+\lambda_2^{\Phi(w_{i-2},w_{i-1},w_i)} \times q_{ML}(w_i \mid w_{i-1}) \\ &+\lambda_3^{\Phi(w_{i-2},w_{i-1},w_i)} \times q_{ML}(w_i) \end{aligned}$$

Notice that we now have 12 smoothing parameters, i.e., λ_j^i for $i = 1 \ldots 4$ and $j = 1 \ldots 3$.

Question: Unfortunately this estimation method has a serious problem: what is it?

Solution

Recall, in class (slide 17, lecture 2), we stated that we must choose values for $\lambda_1, \lambda_2, \lambda_3$ such that $\lambda_1 + \lambda_2 + \lambda_3 = 1$, but

$$
\begin{aligned}
\sum_{w_i} q(w_i \mid w_{i-2}, w_{i-1}) &= \sum_{w_i} [\lambda_1^{\Phi(w_{i-2},w_{i-1},w_i)} \times q_{ML}(w_i \mid w_{i-2}, w_{i-1}) \\
&\quad + \lambda_2^{\Phi(w_{i-2},w_{i-1},w_i)} \times q_{ML}(w_i \mid w_{i-1}) \\
&\quad + \lambda_3^{\Phi(w_{i-2},w_{i-1},w_i)} \times q_{ML}(w_i)] \\
&= \sum_{w_i} \lambda_1^{\Phi(w_{i-2},w_{i-1},w_i)} \times q_{ML}(w_i \mid w_{i-2}, w_{i-1}) \\
&\quad + \sum_{w_i} \lambda_2^{\Phi(w_{i-2},w_{i-1},w_i)} \times q_{ML}(w_i \mid w_{i-1}) \\
&\quad + \sum_{w_i} \lambda_3^{\Phi(w_{i-2},w_{i-1},w_i)} \times q_{ML}(w_i)
\end{aligned}
$$

We are now stuck at this step because λ is dependent on $\sum_{w_i}$ so it cannot be extracted from the equation. Therefore, we cannot guarantee that $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

Question 3 (15 points)

We are going to come up with a modified version of the Viterbi algorithm for trigram taggers. Assume that the input to the Viterbi algorithm is a word sequence $x_1 \dots x_n$. For each word in the vocabulary, we have a *tag dictionary* $T(x)$ that lists the tags y such that $e(x|y) > 0$. Take K to be a constant such that $|T(x)| \leq K$ for all x. Give pseudo-code for a version of the Viterbi algorithm that runs in $O(nK^3)$ time where n is the length of the input sentence.

Solution

Instead of looking at all tags in $\mathcal{K}$, we look at the subset of tags in the words dictionary, $T(x)$. The algorithm is modified as follows.

Input: a sentence $x_1 \dots x_n$, parameters $q(s|u, v)$, $e(x|s)$, and $T(x_1) \dots T(x_n)$.

Initialization: Set $\pi(0, *, *) = 1$, and $\pi(0, u, v) = 0$ for all (u, v) such that $u \neq *$ or $v \neq *$.

Algorithm:

- For $k = 1 \dots n$,
  - For $u \in T(x_{k-1})$, $v \in T(x_k)$,

$$\pi(k, u, v) = \max_{w \in T(x_{k-2})} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$

- **Return** $\max_{u \in T(x_{n-1}), v \in T(x_n)} (\pi(n, u, v) \times q(\text{STOP}|u, v))$

  The runtime of the original Viterbi algorithm is $O(n\mathcal{K}^3)$. Since we are looking at a subset of the tags, T(x), where $|T(x)| \leq K$ the runtime for the modified Viterbi algorithm is $O(nK^3)$.