NLP - 097215
Computer Excersize No. 1

Avrech Ben-David - 200452282
Ilan Frank - 043493386

December 25, 2018

1 Model No. 1

1.1 Overview

We implemented Maximum-Entropy Markov Model (MEMM) as we taught in class. The MEMM model is a feature based discriminative model, which assigns a probability to a full sentence tagging, factorizing the joint probability as follows:

$$P(y_1, y_2, ..., y_n|x; v) = \prod_{i=1}^{n} P(y_i|y_1, ..., y_{i-1}, x; v) \tag{1}$$

The probability of tagging the word x_i by y_i is computed by the softmax term:

$$P(y_i|y_1, ..., y_{i-1}, x; v) = \frac{e^{f(x_i,y_i)\cdot v}}{\sum\limits_{y'\in Y} e^{f(x_i,y')\cdot v}} \tag{2}$$

Where $f(x_i, y_i)$ is a featurizing function, which takes the context of the word x_i (considering the whole sentence, and the previos words' tags) and calculate a binari feature vector.

The feature vector of each tag proposal is weighted by the model's parameter vector v. The parameters v are learned upon the training-set in order to maximize the log-likelyhood of the correct tagging for each word-tag pair in the corpus S. TODO - AM I RIGHT???????????

$$v = \underset{v'}{\text{argmax}}\, log\left(\prod_{(x_i,y_i)\in S} \frac{e^{f(x_i,y_i)\cdot v'}}{\sum\limits_{y'\in Y} e^{f(x_i,y')\cdot v'}}\right) \tag{3}$$

$$= \underset{v'}{\text{argmax}} \sum_{(x_i,y_i)\in S} log\left(\frac{e^{f(x_i,y_i)\cdot v'}}{\sum\limits_{y'\in Y} e^{f(x_i,y')\cdot v'}}\right) \tag{4}$$

$$= \underset{v'}{\text{argmax}} \sum_{(x_i,y_i)\in S} f(x_i, y_i)\cdot v' - log\left(\sum_{y'\in Y} e^{f(x_i,y')\cdot v'}\right) \tag{5}$$

If we want to prevent overflow in the exponent calculations, we have to use the safe softmax version $SafeSoftmax(\vec{x}) = Softmax(\vec{x} - max(\vec{x}))$:

$$v = \underset{v'}{\text{argmax}} \sum_{(x_i,y_i)\in S} log\left(\frac{e^{f(x_i,y_i)\cdot v' - \max_{y'} f(x_i,y')\cdot v'}}{\sum\limits_{y'\in Y} e^{f(x_i,y')\cdot v' - max_{y'} f(x_i,y')\cdot v'}}\right) \tag{6}$$

$$= \underset{v'}{\text{argmax}} \sum_{(x_i,y_i)\in S} f(x_i, y_i)\cdot v' - max_{y'} f(x_i, y')\cdot v' - log\left(\sum_{y'\in Y} e^{f(x_i,y')\cdot v' - max_{y'} f(x_i,y')\cdot v'}\right) \tag{7}$$

The model is composed of the following parts:

- Data-preprocessing, and feature set selection.
- Training - Parameter vector optimization.
- Inference - A modified Viterbi algorithm.

After describing our considerations of the model implementation in detail, we show our model performance on the test-set.

1.2 Data Preprocessing and Feature-Set Selection

2 Question 1.

Define a trigram model:

$$q(w_i \mid w_{i-1}, w_{i-2}) = \lambda_1 q_{ML}(w_i \mid w_{i-1}, w_{i-2}) + \lambda_2 q_{ML}(w_i \mid w_{i-1}) + \lambda_3 q_{ML}(w_i) \tag{8}$$

$S = \{s_1, s_2, ..., s_N\}$ - a validation set consists of N sentences. The number of appearances of a trigram in the validation set:

$$c'(w_1, w_2, w_3) \tag{9}$$

An objective function:

$$L(\lambda_1, \lambda_2, \lambda_3) = \sum_{\{w_1,w_2,w_3\} \in S} c'(w_1, w_2, w_3) log(q(w_i \mid w_{i-1}, w_{i-2})) \tag{10}$$

The perplexity score:

$$l = \frac{1}{N} \sum_{s_i \in S} log(p(s_i)) \tag{11}$$

$$p(q) = 2^{-l} \tag{12}$$

Prove that maximizing L on the validation set is equal to minimizing the perplexity score.

Proof. Minimizing the perplexity score:

$$\underset{\{\lambda_1,\lambda_2,\lambda_3\}}{\text{argmin}}\ p(q) = \underset{\{\lambda_1,\lambda_2,\lambda_3\}}{\text{argmin}}\ 2^{-l} \tag{13}$$

$$= \underset{\{\lambda_1,\lambda_2,\lambda_3\}}{\text{argmax}}\ \frac{1}{N} \sum_{s_i \in S} log(p(s_i)) \tag{14}$$

$$= \underset{\{\lambda_1,\lambda_2,\lambda_3\}}{\text{argmax}} \sum_{s_i \in S} log(p(s_i)) \tag{15}$$

The probability of a full sentence is factorized according to the trigram model:

$$= \underset{\{\lambda_1,\lambda_2,\lambda_3\}}{\operatorname{argmax}} \sum_{s_i \in S} log \prod_{w_j \in s_i} q(w_j \mid w_{j-1}, w_{j-2}) \tag{16}$$

$$= \underset{\{\lambda_1,\lambda_2,\lambda_3\}}{\operatorname{argmax}} \sum_{s_i \in S} \sum_{w_j \in s_i} log(q(w_j \mid w_{j-1}, w_{j-2})) \tag{17}$$

The double sum in (10) can be replaced by a single sum, which sums the probability of the trigrams over the entire validation set, word by word:

$$= \underset{\{\lambda_1,\lambda_2,\lambda_3\}}{\operatorname{argmax}} \sum_{w_j \in S} log(q(w_j \mid w_{j-1}, w_{j-2})) \tag{18}$$

By definition, each trigram is summed exactly $c'(w_1, w_2, w_3)$ times. So instead of summing over the words in the validation set as in (10), we should sum over the trigrams themselves, and weigh each trigram by its appearance counter:

$$= \underset{\{\lambda_1,\lambda_2,\lambda_3\}}{\operatorname{argmax}} \sum_{\{w_j, w_{j-1}, w_{j-2}\} \in S} c'(w_1, w_2, w_3) log(q(w_j \mid w_{j-1}, w_{j-2})) \tag{19}$$

This is exactly the term of the objective function. $\square$

Question 2.

The problem with the proposed estimation method is that it does not solve the case of zero counts. The maximum likelihood estimator is still undefined if a the bigram in it's denominator is zero. This smoothing method improves the prediction only for bigrams which have been seen in the training set.

Another problem is evaluating the lambdas. The described function is not differential, so there is not a simple way to find the optimal values (say using gradient descent).

Question 3.

In the standard Viterbi we take the tag sets to be the set of all tags. In this case we can use our prior knowledge to limit each set S_k to $T(x_k)$.

```
**Input:** A sentence $x_1, .., x_n$, parameters $q(s|u, v)$ and $e(x|s)$
Set $\pi(0, *, *)$
Define $S_{-1} = S_0 = \{*\}$, $S_k = T(x_k)$ for $k \in \{1, .., n\}$
for k=1..n do
    for $u \in S_{k-1}, v \in S_k$ do
        $\pi(k, u, v) = \max_{w \in S_{k-2}} \pi(k-1, w, u) q(v|w, u) e(x_k|v)$
        $bp(k, u, v) = arg\,max_{w \in S_{k-2}} \pi(k-1, w, u) q(v|w, u) e(x_k|v)$
    end
end
Set $(y_{n-1}, y_n) = arg\,max_{u,v} \pi(n, u, v) q(STOP|u, v)$
for $k = (n-2)..1$ do
    $y_k = bp(k+2, y_{k+1}, y_{k+2})$
end
return $y_1, .., y_n$
```

By limiting the tags sets in the main loop from S to $T(x_i)$, we change the Viterbi running time from $O(n|S|^3)$ to $O(n|K|^3)$

Question 4.

Define a trigram model as follows:

$$p(\vec{w}) \overset{\text{def}}{=} p(w_1)p(w_2 \mid w_1)p(w_2 \mid w_1, w_2) \cdots p(w_n \mid w_{n-2}, w_{n-1}) \tag{20}$$

1.

Lets expand (13) using the naive estimates:
$p(w_n \mid w_{n-2}, w_{n-1}) \overset{\text{def}}{=} \dfrac{c(w_{n-2}, w_{n-1}, w_n)}{c(w_{n-2}, w_{n-1})}$

$$p(\vec{w}) = \frac{c(w_1)}{c()} \frac{c(w_1, w_2)}{c(w_1)} \frac{c(w_1, w_2, w_3)}{c(w_1 w_2)} \frac{c(w_2, w_3, w_4)}{c(w_2 w_3)} \cdots \frac{c(w_{n-2}, w_{n-1}, w_n)}{c(w_{n-2}, w_{n-1})} \tag{21}$$

2.

Define the reversed trigram model:

$$p_{reversed}(\vec{w}) \overset{\text{def}}{=} p(w_n)p(w_{n-1} \mid w_n)p(w_{n-2} \mid w_n, w_{n-1}) \cdots p(w_1 \mid w_3, w_2) \tag{22}$$

Lets expand the term above using the naive ML estimates:

$$p_{reversed}(\vec{w}) = \frac{c(w_n)}{c()} \frac{c(w_{n-1}, w_n)}{c(w_n)} \cdots \frac{c(w_1, w_2, w_3)}{c(w_2, w_3)} \tag{23}$$

We get:

$$\begin{aligned}
\bar{p}(\vec{w}) &= \frac{c(w_1)}{c()}\frac{c(w_1,w_2)}{c(w_1)}\frac{c(w_1,w_2,w_3)}{c(w_1w_2)}\frac{c(w_2,w_3,w_4)}{c(w_2w_3)}\cdots\frac{c(w_{n-2},w_{n-1},w_n)}{c(w_{n-2},w_{n-1})} && (24)\\
&= \frac{c(w_1,w_2,w_3)}{c()}\frac{c(w_2,w_3,w_4)}{c(w_2w_3)}\cdots\frac{c(w_{n-2},w_{n-1},w_n)}{c(w_{n-2},w_{n-1})} && (25)\\
&= \frac{c(w_1,w_2,w_3)}{c(w_2w_3)}\frac{c(w_2,w_3,w_4)}{c(w_3,w_4)}\cdots\frac{c(w_{n-2},w_{n-1},w_n)}{c(w_{n-1},w_n)}\frac{c(w_{n-1},w_n)}{c(w_n)}\frac{c(w_n)}{c()} && (26)\\
&= \bar{p}_{reversed}(\vec{w}) && (27)
\end{aligned}$$

3.

The parameter which prevents ending a sentence with "the" is the last factor in the multiplication term - $\frac{c(w_{n-1},w_n,<\backslash n>)}{c(w_{n-1},w_n)}$. The word "the" probably was never seen in the end of any sentence in the training data, so the count term $c(w_{n-1},w_n,<\backslash n>)$ will be equal to zero. In fact we smooth the probability, so we won't get identical zero, but the matter is done.