1 Supervised POS Tagging

1.1 MEMMs and adding features in HMMs

1. As a reminder, in Hidden Markov Models (HMM) for POS Tagging, states are tags and words are observations.

2. The HMM is a generative model. To estimate the probability of a tag sequence given a word sequence (posterior probability), we use Bayes rule and estimate instead the probability of associating the word to the given tag (likelihood probability, corresponding to the emission matrix of the HMM) and the probability of the current tag given the previous tag(s) (prior probability, corresponding to the transition matrix of the HMM).

\[
P(t^n | w^n) \approx \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1}) \rightarrow \text{Bigram POS Tagger.}
\]

\[
P(t^n | w^n) \approx \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1} t_{i-2}) \rightarrow \text{Trigram POS Tagger.}
\]

3. The learning process for a HMM is quite fast because the emission matrix and the transition matrix are calculated by simple frequency counting. However, it becomes a problem when new features need to be added.

4. One enhancement to HMM is to add features the emission matrix and turn it into a discriminative classifier. This is known as a **Maximum Entropy Markov Model** (MEMM).

5. At inference time, it is quite efficient to work with these models. The learning process, however, is considerably more difficult because iterative techniques such as gradient descent must be used.

1.2 Label bias problem

1. A problem with all (or many) locally-normalized models is that all the weight that comes into a state is distributed completely to all the successive states. This is known as **label bias**.

2. Label bias affects the training process of Markov Models because the selection of a state is biased towards those with fewer outgoing transitions, since the successive states compete with one another in a local context.
From the example above the, when the current state is S1, the most likely next state to transit to is S2, and from S2 it is most likely to stay in S2. So from this we infer that the most likely sequence selected would be S1, S2, S2, S2 with a probability \( P(S1 \rightarrow S2 \rightarrow S2 \rightarrow S2) = 0.6 \times 0.3 \times 0.3 = 0.054 \), but the best possible sequence comes out to be S1 \( \rightarrow \) S1 \( \rightarrow \) S1 \( \rightarrow \) S1 with a probability of \( P(S1 \rightarrow S1 \rightarrow S1 \rightarrow S1) = 0.4 \times 0.45 \times 0.5 = 0.09 \). Since the sum of all outgoing transition probabilities for all states should sum to 1. So the states with fewer outgoing transitions are favoured. So, the main cause is local normalization.

3. Models like **Conditional Random Fields** avoid this problem by only normalizing scores globally.

### 1.3 Conditional Random Fields (CRFs)

1. Not affected by the label bias problem: normalization occurs globally.

2. The intuition is that the probability of the entire sequence is the one that needs to be maximized, which won’t necessarily be achieved by optimizing the probabilities locally along the way.

3. In this model, the parameter \( Z \) is used as a partition function or global normalizer, which allows to sum over all possible states. The \( Z \) is the sum over all possible states:

\[
Z(w) = \sum_{s_1, \ldots, s_n} \prod_{i} \psi(s_i, s_{i-1}, w)
\]

4. So now \( P(s|w) \) can be computed as follows

\[
P(s|w) = \frac{\prod_{i} \psi(s_i, s_{i-1}, w)}{Z(w)}
\]
5. The normalizer $Z$ is just a constant value. So to find the most likely sequence of tags:

$$argmax_s P(s|w) = argmax_s \frac{\prod_i \psi(s_i, s_{i-1}, w)}{Z(w)} = argmax_s \prod_i \psi(s_i, s_{i-1}, w)$$

6. The edge weights from a given state to its successors do not need to sum to one, but the Viterbi algorithm can still be used to compute the MLE.

7. In general, CRFs are very similar to MaxEnt models. However, they can be more difficult to train since probabilities need to be used and updated frequently along the process, requiring many invocations of the Viterbi algorithm.

So as shown now the sum of all outgoing edge weights for a state doesn’t need to sum to 1.

8. The parameter learning process for the CRF is defined by the following expression:

$$\ell = \sum_{m=1}^{M} \log P(s^{(m)}|w^{(m)}, \alpha)$$

$$= \sum_{m=1}^{M} \log \left( \prod_i \frac{\exp \left( \sum_k \alpha_{ik} f_k(s_i^{(m)}), s_{i-1}^{(m)}, w^{(m)} \right)}{Z(w^{(m)})} \right)$$

$$= \sum m \sum i \sum k \left( \alpha_{ik} f_k(s_i^{(m)}, s_{i-1}^{(m)}, w^{(m)}) - \log Z(w^{(m)}) \right)$$

where the $\alpha_{ik}$ are the parameters of the model.
9. Taking the partial derivative of this expression with respect to $\alpha_{jl}$:

$$\frac{\partial \ell(\alpha)}{\partial \alpha_{jl}} = \sum_m f_l(s_j^{(m)}, s_{j-1}^{(m)}, w^{(m)}) - \frac{\partial}{\partial \alpha_{jl}} \log Z(w^{(m)})$$

$$= \sum_m f_l(s_j^{(m)}, s_{j-1}^{(m)}, w^{(m)}) - \sum_m f_l(s_j, s_{j-1}, w) P(s_j | w^{(m)}, \alpha)$$

which turns out to be an expression similar to the one obtained for MaxEnt, only that computing the probability now requires running inference.

1.4 Comparison between HMMs and CRFs

1. Computing the probability of a given sequence of states is more difficult using a CRF than a HMM because it involves the additional calculation of the normalizer.

2. Finding the most likely sequence of states is roughly as difficult for both methods.

3. The learning process is harder for CRFs. Since there is no closed-form for the minimization of the objective function, its derivative must be approximated using gradient descent or a similar strategy.

4. To summarize, the computational trade-offs between CRFs and HMMs are as follows: while both of them have similar costs when it comes to inference, CRFs are much more expensive to train.

1.5 Conclusions

1. Part of Speech Tagging is very important for information extraction tasks such as identification of noun phrases.

2. Sequence models are good at these sorts of tasks because they're able to represent the necessary context to infer the sense of a word.

3. Generative models tend to have higher asymptotic error, but they approach it faster than discriminative ones.

4. Discriminative models are more flexible for incorporating features because they make no assumptions of independence in the observations.

5. For either kind of model, the accuracy degrades significantly for out of domain data sets and most errors are usually made on the most important things of the domain.

2 Unsupervised POS Tagging

2.1 Generalities

1. When the input data is not labeled, Unsupervised POS Tagging, also known as Part Of Speech Induction, must be used.

2. This technique is very useful because unlabeled data is more easily acquired than labeled one.

3. The main task is to take raw sentences as input and generate labeled sentences as output.
4. Sentence clustering is a related (much easier) unsupervised learning problem, where sentences are grouped into clusters according to a predefined criteria.

5. The objective function (log likelihood) for the sentence clustering problem is defined as

\[
l(\theta) = \log \left( \prod_{m=1}^{M} P(x^{(m)}|\theta) \right)
\]

\[
= \log \left( \prod_{m=1}^{M} \sum_{c=1}^{C} P(x^{(m)}, c|\theta) \right)
\]

\[
= \sum_{m=1}^{M} \log \left( \sum_{c=1}^{C} P(x^{(m)}, c|\theta) \right)
\]

6. Clustering is a well-known unsupervised learning technique for problems where labels are not observed and therefore marginalizing over them is required. In such problems the resulting probabilistic models are non-concave and with local optima.

### 2.2 Challenges

1. One of the main challenges in unsupervised learning is the selection of the criteria for clustering. Given that the the exact criteria is not provided explicitly for each example, the model could learn all kinds of clustering patterns.

2. The advantage of supervised learning is that with enough data and regularization, a large number of features can be specified and the model will assign low weights to the features considered irrelevant for the target class. In contrast, the learning target of unsupervised problems is highly affected by the selection of features.

3. Higher likelihoods do not necessarily correspond to higher accuracies because there may be multiple, non-identifiable values of the latent variables that produce the highest likelihood, or because even when the solution is unique it may not correspond with the actual metric of interest.

4. Because of this, unsupervised models are subject to a lot of tweaking and tuning to design objectives such that higher likelihoods align with better assignments.

5. The time and effort dedicated to tweaking the unsupervised algorithm might be equivalent to manually labeling some of the data in order to use a supervised algorithm.

6. The amount and availability of data are inversely proportional to its level of detailed linguistic annotation, so unsupervised data tends to be more abundant, whereas labeled data is harder to obtain in large amounts.

7. The performance of a Unsupervised POS Tagger can be improved by enforcing some restrictions upon the data e.g., restricting the set of possible tags for a word using a dictionary, or enforcing soft constraints like expecting at least one verb per sentence and punctuation at the end of the sentence.

### 2.3 Merialdo - A discouraging experiment

In this experiment, the intention was to learn from few examples instead of learning from the whole dataset. The setup is as follows:
1. We know the set of allowable tags for each word.

2. We fix the number of training examples (say k) and learn \( P(w|t) \) and \( P(t|t_{-1}, t_{-2}) \) from these training examples.

3. After training on k examples, they are re-estimated using Expectation Maximization.

The observed thing about this experiment is as the number k is increased the accuracy of the model increases and the best is observed when the whole dataset is taken instead of k training examples. So no point in taking few data points for training.

Table 1: Effect of varying k.

<table>
<thead>
<tr>
<th>Iter</th>
<th>( k=0 )</th>
<th>( k=100 )</th>
<th>( k=2000 )</th>
<th>( k=5000 )</th>
<th>( k=10000 )</th>
<th>( k=20000 )</th>
<th>( k=\text{whole dataset} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>77</td>
<td>90</td>
<td>95.4</td>
<td>96.2</td>
<td>96.6</td>
<td>96.9</td>
<td>97.0</td>
</tr>
<tr>
<td>5</td>
<td>84.8</td>
<td>92.9</td>
<td>95.1</td>
<td>95.4</td>
<td>95.6</td>
<td>95.8</td>
<td>95.8</td>
</tr>
<tr>
<td>10</td>
<td>86.6</td>
<td>92.6</td>
<td>94.4</td>
<td>94.8</td>
<td>95.0</td>
<td>95.2</td>
<td>95.2</td>
</tr>
</tbody>
</table>

2.4 Learning: Expectation Maximization (EM)

1. The unsupervised POS Tagging problem can be defined as follows:

\[
l(\theta) = \sum_{m=1}^{M} \log \left( \sum_{s} P(w^{(m)}, s|\theta) \right)
= \sum_{m=1}^{M} \log \left( \sum_{s} Q^{(m)}(s) \frac{P(w^{(m)}, s|\theta)}{Q^{(m)}(s)} \right)
\geq \sum_{m=1}^{M} \sum_{s} \log \left( Q^{(m)}(s) \frac{P(w^{(m)}, s|\theta)}{Q^{(m)}(s)} \right) \quad \text{(Jensen’s Inequality)}
\]

Where M is the number of training examples and \( Q(s) \) is some distribution over the states for a given example.

2. Since this is a non-convex problem with local optima, an alternation between E-M steps must be performed iteratively to increase the objective. For a given set of parameters \( \theta \), the E-step computes the best value for the \( Q(s) \) distribution, followed by an M-step which uses the value of \( Q(s) \) to compute a new set of parameters \( \theta \) that maximizes the weighted likelihood.

3. The M step calculates the parameters \( \theta \) with the following expression:

\[
\theta = \arg\max_{\theta} \sum_{m=1}^{M} \sum_{s} Q^{(m),t}(s) \log \left( \frac{P(w^{(m)}, s|\theta)}{Q^{(m)}(s)} \right)
= \arg\max_{\theta} \sum_{m=1}^{M} \sum_{s} \left( Q^{(m),t}(s) \log P(w^{(m)}, s|\theta) - Q^{(m),t}\log Q^{(m)}(s) \right)
= \arg\max_{\theta} \sum_{m=1}^{M} \sum_{s} \left( Q^{(m),t}(s) \log P(w^{(m)}, s|\theta) \right)
\]

Where t is the number of the iteration. Notice that the constant part in terms of \( \theta \) is ignored, and that the remaining term is nothing but a weighted version of the likelihood, and therefore it is concave.
4. Following, the E step must calculate the best distribution $Q(s)$:

$$Q(s) = \arg\max Q \sum_{m=1}^{M} \sum_s Q^{(m)}(s) \log \left( \frac{P(w^{(m)}, s|\theta^t)}{Q^{(m)}(s)} \right)$$

$$= \arg\max Q \sum_{m=1}^{M} \sum_s Q^{(m)}(s) \log \left( \frac{P(w^{(m)}|\theta^t)P(s|w^{(m)}, \theta^t)}{Q^{(m)}(s)} \right)$$

$$= \arg\max Q \sum_{m=1}^{M} \sum_s Q^{(m)}(s) \log P(w^{(m)}|\theta^t) + \sum_{m=1}^{M} \sum_s Q^{(m)}(s) \log \left( \frac{P(s|w^{(m)}, \theta^t)}{Q^{(m)}(s)} \right)$$

$$= \arg\max Q \sum_{m=1}^{M} P(w^{(m)}|\theta^t) - \sum_{m=1}^{M} KL(Q^{(m)}(s)||P(s|w^{(m)}))$$

Since the KL divergence is non-negative, it can be minimized by setting $Q^{(m)}(s)$ to be the posterior distribution $P(s|w^{(m)})$.

5. With this, the E-M steps that will be performed iteratively are as follows:

$$Q^{(m)}(s) = P(s|w^{(m)}) \quad \text{E step}$$

$$\arg\max_{\theta} \sum_{m=1}^{M} \sum_s Q^{(m),t}(s) \log P(w^{(m)}, s|\theta) \quad \text{M step}$$