Natural Language Understanding
Lecture 10: Introduction to Unsupervised Part-of-Speech Tagging

Frank Keller

School of Informatics
University of Edinburgh
keller@inf.ed.ac.uk

Based on slides by Sharon Goldwater

February 27, 2017
1 Unsupervised Part-of-Speech Tagging

2 Background
   - Hidden Markov Models
   - Expectation Maximization

3 Bayesian HMM
   - Bayesian Estimation
   - Dirichlet Distribution
   - Bayesianizing the HMM
   - Evaluation

Reading: Goldwater and Griffiths (2007).
Background: Jurafsky and Martin (2009: Ch. 6.1–6.4).
**Part-of-Speech Tagging**

**Task:** take a sentence, assign each word a label indicating its syntactic category (part of speech).

Example:

<table>
<thead>
<tr>
<th>NNP</th>
<th>NNP</th>
<th>,</th>
<th>RB</th>
<th>RB</th>
<th>,</th>
<th>VBZ</th>
<th>RB</th>
<th>VB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Campbell</td>
<td>Soup</td>
<td>,</td>
<td>not</td>
<td>surprisingly</td>
<td>,</td>
<td>does</td>
<td>n’t</td>
<td>have</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DT</th>
<th>NNS</th>
<th>TO</th>
<th>VB</th>
<th>IN</th>
<th>DT</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>any</td>
<td>plans</td>
<td>to</td>
<td>advertise</td>
<td>in</td>
<td>the</td>
<td>magazine</td>
</tr>
</tbody>
</table>

Uses Penn Treebank PoS tag set.
Penn Treebank PoS Tagset

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>Determiner</td>
</tr>
<tr>
<td>IN</td>
<td>Preposition or subord. conjunction</td>
</tr>
<tr>
<td>NN</td>
<td>Noun, singular or mass</td>
</tr>
<tr>
<td>NNS</td>
<td>Noun, plural</td>
</tr>
<tr>
<td>NNP</td>
<td>Proper noun, singular</td>
</tr>
<tr>
<td>RB</td>
<td>Adverb</td>
</tr>
<tr>
<td>TO</td>
<td>to</td>
</tr>
<tr>
<td>VB</td>
<td>Verb, base form</td>
</tr>
<tr>
<td>VBZ</td>
<td>Verb, 3rd person singular present</td>
</tr>
</tbody>
</table>

Total of 36 tags, plus punctuation. English-specific.
Current PoS taggers are highly accurate (97% accuracy on Penn Treebank). But they require *manually labelled* training data, which for many major language is not available. Examples:

<table>
<thead>
<tr>
<th>Language</th>
<th>Speakers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Punjabi</td>
<td>109M</td>
</tr>
<tr>
<td>Vietnamese</td>
<td>69M</td>
</tr>
<tr>
<td>Polish</td>
<td>40M</td>
</tr>
<tr>
<td>Oriya</td>
<td>32M</td>
</tr>
<tr>
<td>Malay</td>
<td>37M</td>
</tr>
<tr>
<td>Azerbaijani</td>
<td>20M</td>
</tr>
<tr>
<td>Haitian</td>
<td>7.7M</td>
</tr>
</tbody>
</table>

[From: Das and Petrov, ACL 2011 talk.]

We need models that do not require annotated training data: *unsupervised PoS tagging*. 
All the unsupervised tagging models we will discuss are based on Hidden Markov Models (HMMs).

The parameters of the HMM are $\theta = (\tau, \omega)$. They define:

- $\tau$: the probability distribution over tag-tag transitions;
- $\omega$: the probability distribution over word-tag outputs.
The parameters are sets of \textit{multinomial distributions}. For tag types $t = 1 \ldots T$ and word types $w = 1 \ldots W$:

- $\omega = \omega^{(1)} \ldots \omega^{(T)}$: the output distributions for each tag;
- $\tau = \tau^{(1)} \ldots \tau^{(T)}$: the transition distributions for each tag;
- $\omega(t) = \omega_{1}^{(t)} \ldots \omega_{W}^{(t)}$: the output distribution from tag $t$;
- $\tau(t) = \tau_{1}^{(t)} \ldots \tau_{T}^{(t)}$: the transition distribution from tag $t$.

Goal of this lecture: \textit{introduce clever ways of estimating} $\omega$ \textit{and} $\tau$. 
Example: $\omega^{(\text{NN})}$ is the output distribution for tag NN:

\[
\begin{array}{c}
\omega \\
\hline \\
\text{John} \\
\text{Mary} \\
\text{running} \\
\text{jumping}
\end{array}
\]
Example: $\omega^{(\text{NN})}$ is the output distribution for tag NN:

<table>
<thead>
<tr>
<th>$\omega_{w}^{(\text{NN})}$</th>
<th>$w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>John</td>
</tr>
<tr>
<td>0.0</td>
<td>Mary</td>
</tr>
<tr>
<td>0.2</td>
<td>running</td>
</tr>
<tr>
<td>0.0</td>
<td>jumping</td>
</tr>
</tbody>
</table>
Example: $\omega^{(\text{NN})}$ is the output distribution for tag NN:

<table>
<thead>
<tr>
<th>$\omega^{(\text{NN})}_w$</th>
<th>$w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>John</td>
</tr>
<tr>
<td>0.0</td>
<td>Mary</td>
</tr>
<tr>
<td>0.2</td>
<td>running</td>
</tr>
<tr>
<td>0.0</td>
<td>jumping</td>
</tr>
</tbody>
</table>

Key idea: define priors over the multinomials that are suitable for NLP tasks.
Another way to write the model, often used in statistics and machine learning:

\[ t_i | t_{i-1} = t \sim \text{Multinomial}(\tau(t)) \]
\[ w_i | t_i = t \sim \text{Multinomial}(\omega(t)) \]

This is read as: “Given that \( t_{i-1} = t \), the value of \( t_i \) is drawn from a multinomial distribution with parameters \( \tau(t) \).”
Inference for HMMs

For *inference* (i.e., decoding, applying the model at test time), we need to know $\theta$ and then we can compute $P(t, w)$:

$$P(t, w) = \prod_{i=1}^{n} P(t_i|t_{i-1})P(w_i|t_i) = \prod_{i=1}^{n} \tau_{t_i}^{(t_{i-1})}\omega_{w_i}^{(t_i)}$$

With this, can compute $P(w)$, i.e., a language model:

$$P(w) = \sum_{t} P(t, w)$$

And also $P(t|w)$, i.e., a PoS tagger:

$$P(t|w) = \frac{P(t, w)}{P(w)}$$
Parameter Estimation for HMMs

For estimation (i.e., training the model, determining its parameters), we need a procedure to set $\theta$ based on data.

For this, we can rely on *Bayes Rule*:

$$P(\theta \mid w) = \frac{P(w \mid \theta)P(\theta)}{P(w)} \propto P(w \mid \theta)P(\theta)$$
Maximum Likelihood Estimation

Choose the $\theta$ that makes the data most probable:

$$\hat{\theta} = \arg\max_{\theta} P(w|\theta)$$

Basically, we ignore the prior. In most cases, this is equivalent to assuming a uniform prior.

In supervised systems, the relative frequency estimate is equivalent to the maximum likelihood estimate. In the case of HMMs:

$$\tau_{t'}^{(t)} = \frac{n(t,t')}{n(t)}$$

$$\omega_{w}^{(t)} = \frac{n(t,w)}{n(t)}$$

where $n(e)$ is the number of times $e$ occurs in the training data.
In unsupervised systems, can often use the *expectation maximization* (EM) algorithm to estimate $\theta$:

- **$E$-step:** use current estimate of $\theta$ to compute expected counts of hidden events (here, $n_{(t,t')}$, $n_{(t,w)}$).
- **$M$-step:** recompute $\theta$ using expected counts.

Examples: forward-backward algorithm for HMMs, inside-outside algorithm for PCFGs, k-means clustering.
Maximum Likelihood Estimation

Estimation Maximization sometimes works well:

- word alignments for machine translation;
- anaphora and coreference.

But it often fails:

- probabilistic context-free grammars: highly sensitive to initialization; F-scores reported are generally low;
- for HMMs, even very small amounts of training data have been show to work better than EM;
- similar picture for many other tasks.
We said: to train our model, we need to estimate $\theta$ from the data. But is this really true?

- for language modeling, we estimate $P(w_{n+1}|\theta)$, but what we actually need is $P(w_{n+1}|w)$;
- for PoS tagging, we estimate $P(t|\theta, w)$, but we actually need is $P(t|w)$.
Bayesian Estimation

We said: to train our model, we need to estimate \( \theta \) from the data. But is this really true?

- for language modeling, we estimate \( P(w_{n+1}|\theta) \), but what we actually need is \( P(w_{n+1}|w) \);
- for PoS tagging, we estimate \( P(t|\theta, w) \), but we actually need is \( P(t|w) \).

So we are not actually interested in the value of \( \theta \). We could simply do this:

\[
P(w_{n+1}|w) = \int_{\Delta} P(w_{n+1}|\theta)P(\theta|w) d\theta \tag{1}
\]

\[
P(t|w) = \int_{\Delta} P(t|w, \theta)P(\theta|w) d\theta \tag{2}
\]

*We don’t estimate \( \theta \), we integrate it out.*
Bayesian Integration

This approach is called *Bayesian integration*. Integrating over $\theta$ gives us an *average* over all possible parameters values. Advantages:

- accounts for uncertainty as to the exact value of $\theta$;
- models the shape of the distribution over $\theta$;
- increases robustness: there may be a range of good values of $\theta$;
- we can use priors favoring sparse solutions (more on this later).
Example: we want to predict: will spinner result be “a” or not?

- Parameter $\theta$ indicates spinner result: $P(\theta = a) = .45$, $P(\theta = b) = .35$, $P(\theta = c) = .2$;
- define $t = 1$: result is “a”, $t = 0$: result is not “a”;
- make a prediction about one random variable ($t$) based on the value of another random variable ($\theta$).
Bayesian Integration

Example: we want to predict: will spinner result be “a” or not?

- Parameter $\theta$ indicates spinner result: $P(\theta = a) = .45$, $P(\theta = b) = .35$, $P(\theta = c) = .2$;
- define $t = 1$: result is “a”, $t = 0$: result is not “a”;
- make a prediction about one random variable ($t$) based on the value of another random variable ($\theta$).

**Maximum likelihood approach:** choose most probable $\theta$: $\hat{\theta} = a$, and $P(t = 1|\hat{\theta}) = 1$, so we predict $t = 1$. 
Example: we want to predict: will spinner result be “a” or not?

- Parameter $\theta$ indicates spinner result: $P(\theta = a) = .45$, $P(\theta = b) = .35$, $P(\theta = c) = .2$;
- define $t = 1$: result is “a”, $t = 0$: result is not “a”;
- make a prediction about one random variable ($t$) based on the value of another random variable ($\theta$).

**Maximum likelihood approach:** choose most probable $\theta$: $\hat{\theta} = a$, and $P(t = 1|\hat{\theta}) = 1$, so we predict $t = 1$.

**Bayesian approach:** average over $\theta$:

$$P(t = 1) = \sum_{\theta} P(t = 1|\theta)P(\theta) = 1(.45) + 0(.35) + 0(0.2) = .45,$$

so we predict $t = 0$. 
Choosing the right prior can make integration easier.

This is where the *Dirichlet distribution* comes in. A $K$-dimensional Dirichlet with parameters $\alpha = \alpha_1 \ldots \alpha_K$ is defined as:

$$P(\theta) = \frac{1}{Z} \prod_{j=1}^{K} \theta_j^{\alpha_j-1}$$

We usually only use symmetric Dirichlets, where $\alpha_1 \ldots \alpha_K$ are all equal to $\beta$. We write Dirichlet($\beta$) to mean Dirichlet($\beta, \ldots, \beta$).
A 2-dimensional symmetric Dirichlet($\beta$) prior over $\theta = (\theta_1, \theta_2)$:

$\beta > 1$: prefer uniform distributions
$\beta = 1$: no preference
$\beta < 1$: prefer sparse (skewed) distributions
Bayesianizing the HMM

To Bayesianize the HMM, we augment with it with symmetric Dirichlet priors:

\[ t_i | t_{i-1} = t, \tau^{(t)} \sim \text{Multinomial}(\tau^{(t)}) \]
\[ w_i | t_i = t, \omega^{(t)} \sim \text{Multinomial}(\omega^{(t)}) \]
\[ \tau^{(t)} | \alpha \sim \text{Dirichlet}(\alpha) \]
\[ \omega^{(t)} | \beta \sim \text{Dirichlet}(\beta) \]

To simplify things, we will present a bigram version of the Bayesian HMM; Goldwater and Griffiths (2007) use trigrams.
If we integrate out the parameters $\theta = (\tau, \omega)$, we get:

$$P(t_{n+1}|t, \alpha) = \frac{n(t_n, t_{n+1}) + \alpha}{n(t_n) + T\alpha}$$

$$P(w_{n+1}|t_{n+1}, t, w, \beta) = \frac{n(t_{n+1}, w_{n+1}) + \beta}{n(t_{n+1}) + W_t n_{t+1}\beta}$$

with $T$ possible tags and $W_t$ possible words with tag $t$.

We can use these distributions to find $P(t|w)$ using an estimation method called *Gibbs sampling*.

- use a dictionary that lists possible tags for each word:
  - run: NN, VB, VBN

- the dictionary is actually derived from WSJ corpus;

- train and test on the unlabeled corpus (24,000 words of WSJ):

  53.6% of word tokens have multiple possible tags. Average number of tags per token = 2.3.
Goldwater and Griffiths (2007) evaluate tagging accuracy against the gold-standard WSJ tags and compare to:

- HMM with maximum-likelihood estimation using EM (MLHMM);
- Conditional Random Field with contrastive estimation (CRF/CE).

They also experiment with reducing/eliminating dictionary information.
Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLHMM</td>
<td>74.7</td>
</tr>
<tr>
<td>BHMM ($\alpha = 1$, $\beta = 1$)</td>
<td>83.9</td>
</tr>
<tr>
<td>BHMM (best: $\alpha = .003$, $\beta = 1$)</td>
<td>86.8</td>
</tr>
<tr>
<td>CRF/CE (best)</td>
<td>90.1</td>
</tr>
</tbody>
</table>

- Integrating over parameters is useful in itself, even with uninformative priors ($\alpha = \beta = 1$);
- better priors can help even more, though do not reach the state of the art.
Evaluation: Syntactic Clustering

*Syntactic clustering*: input are the words only, no dictionary is used:

- collapse 45 treebank tags onto smaller set of 17;
- hyperparameters ($\alpha$, $\beta$) are inferred automatically using Metropolis-Hastings sampler;
- standard accuracy measure requires labeled classes, so measure accuracy using best matching of classes.
MLHMM groups instances of the same lexical item together;
BHMM clusters are more coherent, more variable in size.
BHMM transition matrix is sparse, MLHMM is not.
Unsupervised Part-of-Speech Tagging
Background
Bayesian HMM

Summary

- Unsupervised PoS tagging is useful to build lexica and taggers for new language or domains;
- maximum likelihood HMM with EM performs poorly;
- Bayesian HMM with Gibbs sampling can be used instead;
- the Bayesian HMM improves performance by averaging out uncertainty;
- it also allows us to use priors that favor sparse solutions as they occur in language data.

Recent work on unsupervised tagging uses *HMMs with logistic regression* and *HMMs with word embeddings*: next lecture.
