Maximum Entropy Tagging

James Curran and Stephen Clark
University of Edinburgh

August 2003
Outline

- Introduction to tagging
- Language modelling
- Tagging with probabilities
  - Markov Model tagging
- Feature-based tagging
- Maximum Entropy tagging
  - features in maximum entropy models
  - estimating the feature weights
- Named entity tagging
Mr. Vinken is chairman of Elsevier N.V.

<table>
<thead>
<tr>
<th>Mr.</th>
<th>Vinken</th>
<th>is</th>
<th>chairman</th>
<th>of</th>
<th>Elsevier</th>
<th>N.V.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NNP</td>
<td>NNP</td>
<td>VBZ</td>
<td>NN</td>
<td>IN</td>
<td>NNP</td>
<td>NNP</td>
</tr>
<tr>
<td>I-NP</td>
<td>I-NP</td>
<td>I-VP</td>
<td>I-NP</td>
<td>I-PP</td>
<td>I-NP</td>
<td>I-NP</td>
</tr>
<tr>
<td>I-PER</td>
<td>I-PER</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>I-ORG</td>
<td>I-ORG</td>
</tr>
</tbody>
</table>

the Dutch publishing group.

<table>
<thead>
<tr>
<th>the</th>
<th>Dutch</th>
<th>publishing</th>
<th>group</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>NNP</td>
<td>VBG</td>
<td>NN</td>
</tr>
<tr>
<td>I-NP</td>
<td>I-NP</td>
<td>I-NP</td>
<td>I-NP</td>
</tr>
<tr>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Curran/Clark Maximum Entropy Tagging August 2003
Part of Speech (POS) Tagging

Mr. Vinken is chairman of Elsevier N.V.,

the Dutch publishing group.

- 45 POS tags
- 1 million words Penn Treebank WSJ text
- 97% state of the art accuracy
Chunk Tagging

Mr. Vinken is chairman of Elsevier N.V.,
I-NP I-NP I-VP I-NP I-PP I-NP O

the Dutch publishing group .
I-NP I-NP I-NP I-NP O

- 18 phrase tags
- B-XX separates adjacent phrases of same type
- 1 million words Penn Treebank WSJ text
- 94% state of the art accuracy
Named Entity Tagging

Mr. Vinken is chairman of Elsevier N.V.,
I-PER I-PER O O O I-ORG I-ORG O
the Dutch publishing group .
O O O O O O

- 9 named entity tags
- B-XX separates adjacent phrases of same type
- 160,000 words Message Understanding Conference (MUC-7) data
- 92-94% state of the art accuracy
Language Modelling

- Find the best sequence (words, tags, base pairs, \ldots)  
  \[ \arg \max_{y_1 \cdots y_n} p(y_1 \cdots y_n) \]
  \[ \rightarrow \text{the most probable sequence} \]

- Chain rule expansion:

\[ p(y_1 \cdots y_n) = p(y_1)p(y_2|y_1)p(y_3|y_1, y_2) \cdots p(y_n|y_1, \cdots, y_{n-1}) \]

predict \(y_1\)  
predict \(y_2\) given \(y_1\)  
predict \(y_3\) given \(y_1\) and \(y_2\)  
\[ \ldots \]
Markov Assumption

- Each prediction cannot depend on entire history

- Markov model approximation:

\[
p(y_1 \cdots y_n) = p(y_1)p(y_2|y_1)p(y_3|y_1, y_2) \cdots p(y_n|y_1, \ldots, y_{n-1}) \\
\approx p(y_1)p(y_2|y_1)p(y_3|y_2) \cdots p(y_n|y_{n-1})
\]

- Current prediction only based on previous prediction

- In theory can use any fixed length history

- In practice a history of 2 is typically used (for English)
Tagging with Probabilities

- Find the best tag sequence *given the sentence* (conditional probability):
  \[
  \arg\max_{t_1 \ldots t_n} p(t_1 \ldots t_n \mid w_1 \ldots w_n)
  \]

- Alternatively maximise \( p(t_1 \ldots t_n, w_1 \ldots w_n) \) (joint probability):
  \[
  \arg\max_{t_1 \ldots t_n} p(t_1 \ldots t_n \mid w_1 \ldots w_n) = \arg\max_{t_1 \ldots t_n} \frac{p(t_1 \ldots t_n, w_1 \ldots w_n)}{p(w_1 \ldots w_n)} = \arg\max_{t_1 \ldots t_n} p(t_1 \ldots t_n, w_1 \ldots w_n)
  \]

- MaxEnt taggers directly maximise conditional probability
- Markov Model taggers maximise joint probability
Markov Model Tagging

- Maximise the joint probability:

\[ p(t_1 \ldots t_n, w_1 \ldots w_n) = p(t_1 \ldots t_n)p(w_1 \ldots w_n|t_1 \ldots t_n) \]

- Tag sequence probability (first order Markov Model):

\[ p(t_1 \ldots t_n) \approx p(t_1)p(t_2|t_1)p(t_3|t_2) \cdots p(t_n|t_{n-1}) \]

- Word sequence probability (given the tags):

\[ p(w_1 \ldots w_n|t_1 \ldots t_n) \approx p(w_1|t_1)p(w_2|t_2) \cdots p(w_n|t_n) \]

- Using \( p(w_1 \ldots w_n|t_1 \ldots t_n) \) is counter-intuitive but correct since we’re maximising the joint probability
Probability Estimation for Markov Models

- Probabilities are estimated from markedup data
- Estimates are simple relative frequencies:

\[
p(t_i | t_{i-1}) = \frac{\text{count}(t_{i-1}, t_i)}{\text{count}(t_{i-1})}
\]

\[
p(w_i | t_i) = \frac{\text{count}(w_i, t_i)}{\text{count}(t_i)}
\]
Finding the most probable sequence

- Current decision depends on previous decision(s)
- Cannot simply take the most probable tag for each word
- Viterbi algorithm finds the shortest path through the tag lattice
  - $O(n^2)$ in the number of tags (e.g. POS tags $45^2$)
- Beam search works well in practice
  - $O(n^2)$ in the beam width (typically $5^2$)
Problems with Markov Model Taggers

- unreliable zero or very low counts
  - does a zero count indicate an impossible event?
  - $\Rightarrow$ smoothing the counts solves this problem

- Words not seen in the data are especially problematic
  - $\Rightarrow$ would like to include word internal information
    - e.g. capitalisation or suffix information

- Cannot incorporate diverse pieces of evidence for predicting tags
  - e.g. global document information
Feature-based Models

- Features encode evidence from the context for a particular tag:

  (title caps, NNP)  Citibank, Mr.
  (suffix -ing, VBG) running, cooking

  (POS tag DT, I-NP) the bank, a thief
  (current word from, I-PP) from the bank

  (next word Inc., I-ORG) Lotus Inc.
  (previous word said, I-PER) said Mr. Vinken
Complex Features

- Features can be arbitrarily complex
  - e.g. document level features
    (document = cricket & current word = Lancashire, I-ORG)
    \[\text{\rightarrow hopefully tag Lancashire as I-ORG not I-LOC}\]

- Features can be combinations of atomic features
  - (current word = Miss & next word = Selfridges, I-ORG)
    \[\text{\rightarrow hopefully tag Miss as I-ORG not I-PER}\]
Feature-based Tagging

- How do we incorporate features into a probabilistic tagger?

- Hack the Markov Model tagger to incorporate features
  - estimate probabilities directly from feature counts

- Maximum Entropy (MaxEnt) Tagging
  - principled way of incorporating features
  - requires sophisticated estimation method
Unknown Words in Markov Model Tagging

• Calculate $p(w_i | t_i)$ separately for unknown words:

$$p(w_i | t_i) = p(unknown | t_i) \cdot p(caps | t_i) \cdot p(suffix | t_i)$$

• Feature probabilities calculated using relative frequencies

• Assumes independence between features
  $\implies$ does not account for feature interaction

• Cannot incorporate more complex features
Features in Maximum Entropy Models

- Features encode elements of the context $C$ useful for predicting tag $t$
- Features are binary valued functions, e.g.

$$f_i(C, t) = \begin{cases} 
1 & \text{if } \text{word}(C) = \text{Moody} \& \ t = \text{I-ORG} \\
0 & \text{otherwise}
\end{cases}$$

- $\text{word}(C) = \text{Moody}$ is a contextual predicate
- Features determine $(\text{contextual predicate, } \text{tag})$ pairs (as before)
The Model

\[ p(t|C) = \frac{1}{Z(C)} \exp \left( \sum_{i=1}^{n} \lambda_i f_i(C, t) \right) \]

- \( f_i \) is a feature
- \( \lambda_i \) is a weight (large value implies informative feature)
- \( Z(C) \) is a normalisation constant ensuring a proper probability distribution
- Also known as a log-linear model
- Makes no independence assumptions about the features
Tagging with Maximum Entropy Models

- The conditional probability of a tag sequence $t_1 \ldots t_n$ is

$$p(t_1 \ldots t_n | w_1 \ldots w_n) \approx \prod_{i=1}^{n} p(t_i | C_i)$$

given a sentence $w_1 \ldots w_n$ and contexts $C_1 \ldots C_n$

- The context includes previously assigned tags (for a fixed history)

- Beam search is used to find the most probable sequence
Model Estimation

\[ p(t|C') = \frac{1}{Z(C')} \exp \left( \sum_{i=1}^{n} \lambda_i f_i(C', t) \right) \]

- Model estimation involves setting the weight values \( \lambda_i \)

- The model should reflect the data
  \[ \Rightarrow \text{use the data to } \text{constrain} \text{ the model} \]

- What form should the constraints take?
  \[ \Rightarrow \text{constrain the } \text{expected value} \text{ of each feature } f_i \]
The Constraints

\[ E_p f_i = \sum_{C,t} p(C,t) f_i(C,t) = K_i \]

- Expected value of each feature must satisfy some constraint \( K_i \)

- A natural choice for \( K_i \) is the average empirical count:

\[ K_i = E_{\bar{p}} f_i = \frac{1}{N} \sum_{j=1}^{N} f_i(C_j, t_j) \]

derived from the training data \((C_1, t_1), \ldots, (C_N, t_N)\)
Choosing the Maximum Entropy Model

- The constraints do not *uniquely* identify a model

- From those models satisfying the constraints: *choose the Maximum Entropy model*

- The maximum entropy model is the *most uniform model*  
  \[ \rightarrow \] makes no assumptions in addition to what we know from the data

- Set the weights to give the MaxEnt model satisfying the constraints  
  \[ \rightarrow \] use *Generalised Iterative Scaling (GIS)*
Generalised Iterative Scaling (GIS)

- Set $\lambda_i^{(0)}$ equal to some arbitrary value (e.g. zero)

- Repeat until convergence:

$$\lambda_i^{(t+1)} = \lambda_i^{(t)} + \frac{1}{C} \log \frac{E_p f_i}{E_{p(t)} f_i}$$

where

$$C = \max_{x,y} \sum_{i=1}^{n} f_i(x,y)$$
Smoothing

- Models which satisfy the constraints exactly tend to *overfit* the data

- In particular, empirical counts for low frequency features can be unreliable
  - often leads to very large weight values

- Common smoothing technique is to ignore low frequency features
  - but low frequency features may be important

- Use a *prior* distribution on the parameters
  - encodes our knowledge that weight values should not be too large
Gaussian Smoothing

- We use a *Gaussian prior* over the parameters
  - penalises models with extreme feature weights

- This is a form of *maximum a posteriori* (MAP) estimation

- Can be thought of as relaxing the model constraints

- Requires a modification to the update rule