Part Of Speech (POS) Tagging

Based on
“Foundations of Statistical NLP” by C. Manning & H. Schütze, ch. 10
MIT Press, 2002
1. POS Tagging: Overview

- **Task:** labeling (tagging) each word in a sentence with the appropriate POS (morphological category)

- **Applications:** partial parsing, chunking, lexical acquisition, information retrieval (IR), information extraction (IE), question answering (QA)

- **Approaches:**
  - Hidden Markov Models (HMM)
  - Transformation-Based Learning (TBL)
  - others: neural networks, decision trees, bayesian learning, maximum entropy, etc.

- **Performance acquired:** 90% – 98%
Sample POS Tags
(from the Brown/Penn corpora)

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>article</td>
</tr>
<tr>
<td>BEZ</td>
<td>is</td>
</tr>
<tr>
<td>IN</td>
<td>preposition</td>
</tr>
<tr>
<td>JJ</td>
<td>adjective</td>
</tr>
<tr>
<td>JJR</td>
<td>adjective: comparative</td>
</tr>
<tr>
<td>MD</td>
<td>modal</td>
</tr>
<tr>
<td>NN</td>
<td>noun: singular or mass</td>
</tr>
<tr>
<td>NNP</td>
<td>noun: singular proper</td>
</tr>
<tr>
<td>NNS</td>
<td>noun: plural</td>
</tr>
<tr>
<td>PERIOD</td>
<td>.:?!</td>
</tr>
<tr>
<td>PN</td>
<td>personal pronoun</td>
</tr>
<tr>
<td>RB</td>
<td>adverb</td>
</tr>
<tr>
<td>RBR</td>
<td>adverb: comparative</td>
</tr>
<tr>
<td>TO</td>
<td>to</td>
</tr>
<tr>
<td>VB</td>
<td>verb: base form</td>
</tr>
<tr>
<td>VBD</td>
<td>verb: past tense</td>
</tr>
<tr>
<td>VBG</td>
<td>verb: past participle, gerund</td>
</tr>
<tr>
<td>VBN</td>
<td>verb: past participle</td>
</tr>
<tr>
<td>VBP</td>
<td>verb: non-3rd singular present</td>
</tr>
<tr>
<td>VBZ</td>
<td>verb: 3rd singular present</td>
</tr>
<tr>
<td>WDT</td>
<td>wh-determiner (what, which)</td>
</tr>
</tbody>
</table>
An Example

The representative put chairs on the table.

*put* – option to sell; *chairs* – leads a meeting

Tagging requires (limited) **syntactic disambiguation**.

But, there are multiple POS for many words

English has production rules like noun → verb

(e.g., *flour* the pan, *bag* the groceries)

So,...
The first approaches to POS tagging

• [Greene & Rubin, 1971]
  deterministic rule-based tagger
  77% of words correctly tagged — not enough; made the problem look hard

• [Charniak, 1993]
  statistical, “dumb” tagger, based on Brown corpus
  90% accuracy — now taken as baseline
2. POS Tagging Using Markov Models

Assumptions:

- **Limited Horizon:**
  \[ P(t_{i+1} | t_{1,i}) = P(t_{i+1} | t_i) \]
  (first-order Markov model)

- **Time Invariance:**
  \[ P(X_{k+1} = t^j | X_k = t^i) \] does not depend on \( k \)

- **Words are independent of each other**
  \[ P(w_{1,n} | t_{1,n}) = \prod_{i=1}^{n} P(w_i | t_{1,n}) \]

- **A word’s identity depends only of its tag**
  \[ P(w_i | t_{1,n}) = P(w_i | t_i) \]
Determining Optimal Tag Sequences
The Viterbi Algorithm

\[
\text{argmax}_{t_1...n} P(t_1...n|w_1...n) = \text{argmax}_{t_1...n} \frac{P(w_1...n|t_1...n)P(t_1...n)}{P(w_1...n)}
\]

\[
= \text{argmax}_{t_1...n} P(w_1...n|t_1...n)P(t_1...n)
\]

using the previous assumptions

\[
= \text{argmax}_{t_1...n} \prod_{i=1}^{n} P(w_i|t_i) \prod_{i=1}^{n} P(t_i|t_{i-1})
\]

2.1 **Supervised POS Tagging** — using tagged training data:

**MLE estimations:**

\[
P(w|t) = \frac{C(w,t)}{C(t)}, \quad P(t''|t') = \frac{C(t',t'')}{C(t')}
\]
Exercises

10.4, 10.5, 10.6, 10.7, pag 348–350

[Manning & Schütze, 2002]
The Treatment of Unknown Words (I)

• use a priori uniform distribution over all tags: badly lowers the accuracy of the tagger

• feature-based estimation [Weishedel et al., 1993]:
  \[ P(w|t) = \frac{1}{Z} P(\text{unknown word} \mid t)P(\text{Capitalized} \mid t)P(\text{Ending} \mid t) \]
  where \( Z \) is a normalization constant:
  \[ Z = \sum_{t'} P(\text{unknown word} \mid t')P(\text{Capitalized} \mid t')P(\text{Ending} \mid t') \]
  error rate 40% \( \Rightarrow \) 20%

• using both roots and suffixes [Charniak, 1993]
  example: \textit{doe-s} (verb), \textit{doe-s} (noun)
The Treatment of Unknown Words (II)

Smoothing

• (“Add One”) [Church, 1988]

\[ P(w|t) = \frac{C(w, t) + 1}{C(t) + k_t} \]

where \( k_t \) is the number of possible words for \( t \)

• [Charniak et al., 1993]

\[ P(t''|t') = (1 - \epsilon) \frac{C(t', t'')}{C(t')} + \epsilon \]

Note: not a proper probability distribution
2.2 Unsupervised POS Tagging using HMMs

no labeled training data;
use the **EM** (Forward-Backward) algorithm

**Initialisation** options:

- random: not very useful (do $\approx 10$ iterations)
- when a dictionary is available (2-3 iterations)
  - [Jelinek, 1985]
    \[
    b_{j,l} = \frac{b_{j,l}^* C(w^l)}{\sum_{w,m} b_{j,m}^* C(w^m)} \quad \text{where} \quad b_{j,l}^* = \begin{cases} 
    0 & \text{if } t^j \text{ not allowed for } w^l \\
    \frac{1}{T(w^l)} & \text{otherwise}
\end{cases}
\]
  - $T(w^l)$ is the number of tags allowed for $w^l$
  - [Kupiec, 1992] group words into equivalent classes.
    Example:
    \[
    u_{JJ,NN} = \{\text{top, bottom,...}\}, \quad u_{NN,VB,VBP} = \{\text{play, flour, bag,...}\}
    \]
    distribute $C(u_L)$ over all words in $u_L$
2.3 Fine-tuning HMMs for POS Tagging

[ Brands, 1998 ]
Trigram Taggers

- 1st order MMs = bigram models  
  each state represents the previous word’s tag  
  the probability of a word’s tag is conditioned on the previous tag

- 2nd order MMs = trigram models  
  state corresponds to the previous two tags  
  tag probability conditioned on the previous two tags

- example:  
  *is clearly marked* ⇒ BEZ RB VBN more likely than BEZ RB VBD  
  *he clearly marked* ⇒ PN RB VBD more likely than PN RB VBN

- problem: sometimes little or no syntactic dependency, e.g. across commas. Example: *xx, yy: xx* gives little information on *yy*

- more severe data sparseness problem
Linear interpolation

• combine unigram, bigram and trigram probabilities as given by first-order, second-order and third-order MMs on words sequences and their tags

$$P(t_i \mid t_{i-1}) = \lambda_1 P_1(t_i) + \lambda_2 P_2(t_i \mid t_{i-1}) + \lambda_3 P_3(t_i \mid t_{i-1}, i-2)$$

• $\lambda_1, \lambda_2, \lambda_3$ can be automatically learned using the EM algorithm

see [Manning & Schütze 2002, Figure 9.3, pag. 323]
Variable Memory Markov Models

- have states of mixed “length” (instead of fixed length as bigram or trigram tagger have)
- the actual sequence of words/signals determines the length of memory used for the prediction of state sequences
3. POS Tagging based on Transformation-based Learning (TBL) [Brill, 1995]

- exploits a wider range of regularities (lexical, syntactic) in a wider context
- input: tagged training corpus
- output: a sequence of learned transformations rules; each transformation relabels some words
- 2 principal components:
  - specification of the (POS-related) transformation space
  - TBL learning algorithm; transformation selection criterion: greedy error reduction
TBL Transformations

- Rewrite rules: \( t \rightarrow t' \) if condition \( C \)

- Examples:

<table>
<thead>
<tr>
<th>Tag 1</th>
<th>Tag 2</th>
<th>Condition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>VB</td>
<td>previous tag is TO</td>
<td>...try to hammer...</td>
</tr>
<tr>
<td>VBP</td>
<td>VB</td>
<td>one of prev. 3 tags is MD</td>
<td>...could have cut...</td>
</tr>
<tr>
<td>JJR</td>
<td>RBR</td>
<td>next tag is JJ</td>
<td>...more valuable player...</td>
</tr>
<tr>
<td>VBP</td>
<td>VB</td>
<td>one of prev. 2 words in n’t</td>
<td>...does n’t put...</td>
</tr>
</tbody>
</table>

- A later transformation may partially undo the effect.
  Example: *go to school*
TBL POS Algorithm

- tag each word with its most frequent POS
- for $k = 1, 2, ...$
  - Consider all possible transformations that would apply at least once in the corpus
  - set $t_k$ to the transformation giving the greatest error reduction
  - apply the transformation $t_k$ to the corpus
  - stop if termination criterion is met (error rate $< \epsilon$)

- output: $t_1, t_2, ..., t_k$

- issues: 1. search is greedy; 2. transformations applied (lazily...) from left to right
TBL Efficient Implementation:
Using Finite State Transducers [Roche & Scabes, 1995]

\[ t_1, t_2, \ldots, t_n \Rightarrow \text{FST} \]

1. convert each transformation to an equivalent FST: \( t_i \Rightarrow f_i \)

2. create a local extension for each FST: \( f_i \Rightarrow f'_i \)
   so that running \( f'_i \) in one pass on the whole corpus be equivalent to running \( f_i \) on each position in the string

   Example: rule \( A \rightarrow B \) if \( C \) is one of the 2 precedent symbols
   \( CAA \rightarrow CBB \) requires two separate applications of \( f_i \)
   \( f'_i \) does rewrite in one pass

3. compose all transducers: \( f'_1 \circ f'_2 \circ \ldots \circ f'_R \Rightarrow f_{ND} \)
   typically yields a non-deterministic transducer

4. convert to deterministic FST: \( f_{ND} \Rightarrow f_{DET} \)
   (possible for TBL for POS tagging)
TBL Tagging Speed

- **transformations:** $O(Rkn)$
  
  $$R = \text{the number of transformations}$$
  $$k = \text{maximum length of the contexts}$$
  $$n = \text{length of the input}$$
  
- **FST:** $O(n)$ with a much smaller constant
  one order of magnitude faster than a HMM tagger
  
- [André Kempe, 1997] work on HMM $\rightarrow$ FST
Appendix A
Transformation-based Error-driven Learning

Training:
1. unannotated input (text) is passed through an initial state annotator
2. by comparing its output with a standard (e.g. manually annotated corpus), transformation rules of a certain template/pattern are learned to improve the quality (accuracy) of the output.
Reiterate until no significant improvement is obtained.

Note: the algo is greedy: at each iteration, the rule with the best score is retained.

Test:
1. apply the initial-state annotator
2. apply each of the learned transformation rules in order.
Transformation-based Error-driven Learning
Appendix B
Unsupervised Learning of Disambiguation Rules for POS Tagging
[ Eric Brill, 1995 ]

Plan:

1. An unsupervised learning algorithm (i.e., without using a manually tagged corpus) for automatically acquiring the rules for a TBL-based POS tagger

2. Comparison to the EM/Baum-Welch algorithm used for unsupervised training of HMM-based POS taggers

3. Combining unsupervised and supervised TBL taggers to create a highly accurate POS tagger using only a small amount of manually tagged text
1. Unsupervised TBL-based POS tagging

1.1 Start with minimal amount of knowledge:
the allowable tags for each word.

These tags can be extracted from an on-line dictionary or through morphological and distributional analysis.

The “initial-state annotator” will assign all these tags to words in the annotated text.

Example:

Rival/JJ_NNP gangs/NNS have/VB_VBP
turned/VBD_VBN cities/NNS into/IN combat/NN_VB
tzones/NNS ./.
1.2 The transformations which will be learned will reduce the uncertainty. They will have the form:

Change the tag of a word from $X$ to $Y$ in the context $C$.

where $X$ is a set of tags, $Y \in X$, and $C$ is one of the form:

the previous/next tag/word is $T/W$.

Example:

From NN_VB_VBP to VBP if the previous tag is NNS
From NN_VB to VB if the previous tag is MD
From JJ_NNP to JJ if the following tag is NNS
1.3 The scoring

**Note:** While in supervised training the annotated corpus is used for scoring the outcome of applying transformations, in unsupervised training we need an *objective function* to evaluate the effect of learned transformations.

**Idea:** Use information from the distribution of unambiguous words to find reliable disambiguation contexts.

**The value of the objective function:**

The score of the rule

*Change the tag of a word from \( \mathcal{X} \) to \( Y \) in context \( C \).*

is the difference between the number of unambiguous instances of tag \( Y \) in (all occurrences of the context) \( C \) and the number of unambiguous instances of the most likely tag \( R \) in \( C \) (\( R \in \mathcal{X}, R \neq Y \)), adjusting for relative frequency.
Formalisation:
1. Compute:

\[
R = \arg\max_{Z \in \mathcal{X}, \ Z \neq Y} \ \frac{\text{incontext}(Z, C)}{\text{freq}(Z)}
\]

where:

\[
\text{freq}(Z) \text{ is the number of occurrences of words unambiguously tagged } Z \text{ in the corpus;}
\]

\[
\text{incontext}(Z, C) = \text{number of occurrences of words unambiguously tagged } Z \text{ in } C.
\]

Note:

\[
R = \arg\min_{Z \in \mathcal{X}, \ Z \neq Y} \left[ \frac{\text{incontext}(Y, C)}{\text{freq}(Y)} - \frac{\text{incontext}(Z, C)}{\text{freq}(Z)} \right]
\]

where \(\text{freq}(Y)\) is computed similarly to \(\text{freq}(Z)\).
Formalisation (cont’d):

2. The **score** of the (previously) given rule:

\[
\text{incontext}(Y, C) - \frac{\text{incontext}(R, C)}{\text{freq}(R)} = \text{freq}(Y) \left[ \frac{\text{incontext}(Y, C)}{\text{freq}(Y)} - \frac{\text{incontext}(R, C)}{\text{freq}(R)} \right] = \text{freq}(Y) \times \min_{Z \in \mathcal{X}, Z \neq Y} \left[ \frac{\text{incontext}(Y, C)}{\text{freq}(Y)} - \frac{\text{incontext}(Z, C)}{\text{freq}(Z)} \right]
\]

In each iteration the learner searches for the transformation rule which maximizes this score.
1.4 Stop the training when no positive scoring transformations can be found.
2. Unsupervised learning of a POS tagger: Evaluation

2.1 Results
on the Penn treebank corpus [Marcus et al., 1993]: 95.1%
on the Brown corpus [Francis and Kucera, 1982]: 96%

(for more details, see Table 1, page 8 from [Brill, 1995])

2.2 Comparison to the EM/Baum-Welch unsupervised learning:
on the Penn treebank corpus: 83.6%
on 1M words of Associated Press articles: 86.6%;
Kupiec’s version (1992), using classes of words: 95.7%

Note: Compared to the Baum-Welch tagger, no overtraining occurs. (Otherwise an additional held-out training corpus is needed to determine an appropriate number of training iterations.)
3. Weakly supervised rule learning

**Aim:** use a tagged corpus to improve the accuracy of unsupervised TBL.

**Idea:** use the trained unsupervised POS tagger as the “initial-state annotator” for the supervised learner.

**Advantage** over using supervised learning alone:
- use both tagged and untagged text in training.
Combining unsupervised learning and supervised learning
**Difference** w.r.t. weakly supervised Baum-Welch:

- in TBL weakly supervised learning, supervision influences the learner after unsupervised training;
- in weakly supervised Baum-Welch, tagged text is used to bias the initial probabilities.

**Weakness** in weakly supervised Baum-Welch:

unsupervised training may erase what was learned from the manually annotated corpus.

Example: [Merialdo, 1995], 50K tagged words, test accuracy (by probabilistic estimation): 95.4%; but after 10 EM iterations: 94.4%!
**Results:** see Table 2, pag. 11 [ Brill, 1995 ]

**Conclusion:** The combined training outperformed the purely supervised training at no added cost in terms of annotated training text.