Advanced Artificial Intelligence

Part II. Statistical NLP

Applications of HMMs and PCFGs in NLP

Wolfram Burgard, Luc De Raedt, Bernhard Nebel, Lars Schmidt-Thieme

Most slides taken (or adapted) from Adam Przepiorkowski (Poland)
Figures by Manning and Schuetze
Contents

- Part of Speech Tagging
  - Task
  - Why

- Approaches
  - Naive
  - VMM
  - HMM
  - Transformation Based Learning

- Probabilistic Parsing
  - PCFGs and Tree Banks

Parts of chapters 10, 11, 12 of Statistical NLP, Manning and Schuetze, and Chapter 8 of Jurafsky and Martin, Speech and Language Processing.
Motivations and Applications

- Part-of-speech tagging
  - The representative put chairs on the table
  - AT  NN  VBD NNS  IN AT  NN
  - AT  JJ  NN  VBZ  IN AT  NN

- Some tags:
  - AT: article, NN: singular or mass noun, VBD: verb, past tense, NNS: plural noun, IN: preposition, JJ: adjective
<table>
<thead>
<tr>
<th>Tag</th>
<th>Part Of Speech</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>article</td>
</tr>
<tr>
<td>BEZ</td>
<td>the word <em>is</em></td>
</tr>
<tr>
<td>IN</td>
<td>preposition</td>
</tr>
<tr>
<td>JJ</td>
<td>adjective</td>
</tr>
<tr>
<td>JJR</td>
<td>comparative adjective</td>
</tr>
<tr>
<td>MD</td>
<td>modal</td>
</tr>
<tr>
<td>NN</td>
<td>singular or mass noun</td>
</tr>
<tr>
<td>NNP</td>
<td>singular proper noun</td>
</tr>
<tr>
<td>NNS</td>
<td>plural noun</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>comparative adverb</td>
</tr>
<tr>
<td>TO</td>
<td>the word <em>to</em></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, present participle, gerund</td>
</tr>
<tr>
<td>VBN</td>
<td>verb, past participle</td>
</tr>
<tr>
<td>VBP</td>
<td>verb, non-3rd person singular present</td>
</tr>
<tr>
<td>VBZ</td>
<td>verb, 3rd singular present</td>
</tr>
<tr>
<td>WDT</td>
<td><em>wh</em>- determiner (<em>what</em>, <em>which</em>)</td>
</tr>
</tbody>
</table>

Table 10.1 Some part-of-speech tags frequently used for tagging English.
Why pos-tagging?

- First step in parsing
- More tractable than full parsing, intermediate representation
- Useful as a step for several other, more complex NLP tasks, e.g.
  - Information extraction
  - Word sense disambiguation
  - Speech Synthesis
- Oldest task in Statistical NLP
- Easy to evaluate
- Inherently sequential
Different approaches

- Start from tagged training corpus
  - And learn

- Simplest approach
  - For each word, predict the most frequent tag
    - 0-th order Markov Model
    - Gets 90% accuracy at word level (English)

- Best taggers
  - 96-97% accuracy at word level (English)
  - At sentence level: e.g. 20 words per sentence, on average one tagging error per sentence
  - Unsure how much better one can do (human error)
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_i$</td>
<td>the word at position $i$ in the corpus</td>
</tr>
<tr>
<td>$t_i$</td>
<td>the tag of $w_i$</td>
</tr>
<tr>
<td>$w_{i,i+m}$</td>
<td>the words occurring at positions $i$ through $i + m$</td>
</tr>
<tr>
<td></td>
<td>(alternative notations: $w_i \cdots w_{i+m}$, $w_i, \ldots, w_{i+m}$, $w_{i(i+m)}$)</td>
</tr>
<tr>
<td>$t_{i,i+m}$</td>
<td>the tags $t_i \cdots t_{i+m}$ for $w_i \cdots w_{i+m}$</td>
</tr>
<tr>
<td>$w^l$</td>
<td>the $l^{th}$ word in the lexicon</td>
</tr>
<tr>
<td>$t^j$</td>
<td>the $j^{th}$ tag in the tag set</td>
</tr>
<tr>
<td>$C(w^l)$</td>
<td>the number of occurrences of $w^l$ in the training set</td>
</tr>
<tr>
<td>$C(t^j)$</td>
<td>the number of occurrences of $t^j$ in the training set</td>
</tr>
<tr>
<td>$C(t^j, t^k)$</td>
<td>the number of occurrences of $t^j$ followed by $t^k$</td>
</tr>
<tr>
<td>$C(w^l : t^j)$</td>
<td>the number of occurrences of $w^l$ that are tagged as $t^j$</td>
</tr>
<tr>
<td>$T$</td>
<td>number of tags in tag set</td>
</tr>
<tr>
<td>$W$</td>
<td>number of words in the lexicon</td>
</tr>
<tr>
<td>$n$</td>
<td>sentence length</td>
</tr>
</tbody>
</table>

**Table 10.2** Notational conventions for tagging.
Visual Markov Model

- Assume the VMM of last week
- We are representing

\[ P(t^k|t^j) = \frac{C(t^j, t^k)}{C(t^j)} \]

- Lexical (word) information implicit
Table 10.3

<table>
<thead>
<tr>
<th>First tag</th>
<th>AT</th>
<th>BEZ</th>
<th>IN</th>
<th>NN</th>
<th>VB</th>
<th>PERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>48636</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>BEZ</td>
<td>1973</td>
<td>0</td>
<td>426</td>
<td>187</td>
<td>0</td>
<td>38</td>
</tr>
<tr>
<td>IN</td>
<td>43322</td>
<td>0</td>
<td>1325</td>
<td>17314</td>
<td>0</td>
<td>185</td>
</tr>
<tr>
<td>NN</td>
<td>1067</td>
<td>3720</td>
<td>42470</td>
<td>11773</td>
<td>614</td>
<td>21392</td>
</tr>
<tr>
<td>VB</td>
<td>6072</td>
<td>42</td>
<td>4758</td>
<td>1476</td>
<td>129</td>
<td>1522</td>
</tr>
<tr>
<td>PERIOD</td>
<td>8016</td>
<td>75</td>
<td>4656</td>
<td>1329</td>
<td>954</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 10.3  Idealized counts of some tag transitions in the Brown Corpus. For example, NN occurs 48636 times after AT.
Hidden Markov Model

- Make the lexical information explicit and use HMMs
- State values correspond to possible tags
- Observations to possible words
- So, we have

\[
a_{ij} = P(t^j | t^i) \\
b_{ik} = P(w^k | t^i)
\]
Estimating the parameters

- From a **tagged** corpus, maximum likelihood estimation
  
  \[
  a_{ij} = P(t^j | t^i) = \frac{C(t^i, t^j)}{C(t^i)}
  \]

  \[
  b_{ik} = P(w^k | t^j) = \frac{C(w^k : t^j)}{C(t^j)}
  \]

- So, even though a hidden markov model is learning, everything is visible during learning!
- Possibly apply smoothing (cf. N-gramms)
Table 10.4

<table>
<thead>
<tr>
<th></th>
<th>AT</th>
<th>BEZ</th>
<th>IN</th>
<th>NN</th>
<th>VB</th>
<th>PERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>bear</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>43</td>
<td>0</td>
</tr>
<tr>
<td>is</td>
<td>0</td>
<td>10065</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>move</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>36</td>
<td>133</td>
<td>0</td>
</tr>
<tr>
<td>on</td>
<td>0</td>
<td>0</td>
<td>5484</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>president</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>382</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>progress</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>108</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>the</td>
<td>69016</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>48809</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Table 10.4  Idealized counts of tags that some words occur within the Brown Corpus. For example, 36 occurrences of *move* are with the tag NN.
Tagging with HMM

- For an unknown sentence, employ now the Viterbi algorithm to tag
- Similar techniques employed for protein secondary structure prediction

Problems
- The need for a large corpus
- Unknown words (cf. Zipf’s law)
Unknown words

Two classes of part of speech:
open and closed (e.g. articles)
for closed classes all words are known
Z: normalization constant

\[ P(w^i|t^j) = \frac{1}{Z} P(\text{unknown}|t^j) \times P(\text{capitalized}|t^j) \times P(\text{endings}|t^j) \]
What if no corpus available?

- Use traditional HMM (Baum-Welch) but
  - Assume dictionary (lexicon) that lists the possible tags for each word
- One possibility: initialize the word generation (symbol emission) probabilities

\[ b_{jl} = \frac{b^*_j C(w^l)}{\sum_{wm} b^*_j C(w^m)} \]

\[ b^*_j = \begin{cases} 0 & \text{if } t^j \text{ is not a part of speech for } w^l \\ 1 / T(w^l) & \text{otherwise} \end{cases} \]
Assume $b_{jl}^* = P(t^j \mid w^l) = 1 / T(w^l)$, i.e. uniform  

We want  

$$P(w^l \mid t^j) = \frac{P(t^j \mid w^l)P(w^l)}{P(t^j)}$$

$$= \frac{P(t^j \mid w^l)P(w^l)}{\sum_{w^m} P(t^j \mid w^m)P(w^m)}$$

$$= \frac{1.C(w^l)}{T(w^l) \sum_{w^k} C(w^k)}$$

$$= \frac{\sum_{w^m} 1.C(w^m)}{T(w^m) \sum_{w^k} C(w^k)}$$

$$= \frac{C(w^l)}{T(w^l)} \frac{1}{\sum_{w^m} C(w^m)}$$
Transformation Based Learning
(Eric Brill)

- **Observation:**
  - Predicting the most frequent tag already results in excellent behaviour

- **Why not try to correct the mistakes that are made?**
  - Apply transformation rules
    - IF conditions THEN replace tag\_j by tag\_l

- **Which transformations / corrections admissible?**

- **How to learn these?**
Table 10.7  Triggering environments in Brill's transformation-based tagger. Examples: Line 5 refers to the triggering environment “Tag $t^j$ occurs in one of the three previous positions”; Line 9 refers to the triggering environment “Tag $t^j$ occurs two positions earlier and tag $t^k$ occurs in the following position.”
<table>
<thead>
<tr>
<th>Source tag</th>
<th>Target tag</th>
<th>Triggering environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>VB</td>
<td>previous tag is TO</td>
</tr>
<tr>
<td>VBP</td>
<td>VB</td>
<td>one of the previous three tags is MD</td>
</tr>
<tr>
<td>JJR</td>
<td>RBR</td>
<td>next tag is JJ</td>
</tr>
<tr>
<td>VBP</td>
<td>VB</td>
<td>one of the previous two words is <em>n't</em></td>
</tr>
</tbody>
</table>

**Table 10.8** Examples of some transformations learned in transformation-based tagging.
The learning algorithm

1. $C_0 :=$ corpus with each word tagged with its most frequent tag
2. for $k := 0$ step 1 do
3.   $\nu :=$ the transformation $u_i$ that minimizes $E(u_i(C_k))$
4.   if $(E(C_k) - E(\nu(C_k))) < \epsilon$ then break fi
5.   $C_{k+1} := \nu(C_k)$
6.   $\tau_{k+1} := \nu$
7. end
8. Output sequence: $\tau_1, \ldots, \tau_k$

Figure 10.3 The learning algorithm for transformation-based tagging. $C_i$ refers to the tagging of the corpus in iteration $i$. $E$ is the error rate.
Remarks

- Other machine learning methods could be applied as well (e.g. decision trees, rule learning ... )
Rule-based tagging

- Oldest method, hand-crafted rules
- Start by assigning all potential tags to each word
- Disambiguate using manually created rules
- E.g. for the word that
  - If
    - The next word is an adjective, an adverb or a quantifier,
    - And the further symbol is a sentence boundary
    - And the previous word is not a consider-type verb
  - Then erase all tags apart from the adverbial tag
  - Else erase the adverbial tag
Learning PCFGs for parsing

- **Learning from complete data**
  - Everything is “observed” “visible”, examples are parse trees
  - Cf. POS-tagging from tagged corpora
  - PCFGs : learning from tree banks,
  - Easy : just counting

- **Learning from incomplete data**
  - Harder : The EM approach
  - The inside-outside algorithm
  - Learning from the sentences (no parse trees given)
A Penn Treebank tree (POS tags not shown)

(S (NP-SBJ The move)
  (VP followed)
    (NP (NP a round)
      (PP of)
        (NP (NP similar increases)
          (PP by)
            (NP other lenders))
          (PP against)
            (NP Arizona real estate loans)))
)

(S-ADV (NP-SBJ *)
  (VP reflecting)
    (NP (NP a continuing decline)
      (PP-LOC in)
        (NP that market))))
How does it work?

- \( R := \{r \mid r \text{ is a rule that occurs in one of the parse trees in the corpus} \} \)

- For all rules \( r \) in \( R \) do
  - Estimate probability label rule
  - \( P(N \rightarrow S) = \frac{\text{Count}(N \rightarrow S)}{\text{Count}(N)} \)
Conclusions

- Pos-tagging as an application of SNLP
- VMM, HMMs, TBL
- Statistical taggers
  - Good results for positional languages (English)
  - Relatively cheap to build
  - Overfitting avoidance needed
  - Difficult to interpret (black box)
  - Linguistically naive
Conclusions

- **Rule-based taggers**
  - Very good results
  - Expensive to build
  - Presumably better for free word order languages
  - Interpretable

- **Transformation based learning**
  - A good compromise?

- **Tree bank grammars**
  - Pretty effective (and easy to learn)
  - But hard to get the corpus.