Part-of-Speech Tagging

Berlin Chen 2003

References:
1. Speech and Language Processing, chapter 8
2. Foundations of Statistical Natural Language Processing, chapter 10
Review

• Tagging (part-of-speech tagging)
  – The process of assigning (labeling) a part-of-speech or other lexical class marker to each word in a sentence (or a corpus)
    • Decide whether each word is a noun, verb, adjective, or whatever
      The/AT representative/NN put/VBD chairs/NNS on/IN the/AT table/NN
  – An intermediate layer of representation of syntactic structure
    • When compared with syntactic parsing
  – Above 96% accuracy for most successful approaches
Introduction

• Parts-of-speech
  – Known as POS, word classes, lexical tags, morphology classes

• Tag sets
  – Penn Treebank: 45 word classes used (Francis, 1979)
    • Penn Treebank is a parsed corpus
  – Brown corpus: 87 word classes used (Marcus et al., 1993)
  – ....

The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.
## The Penn Treebank POS Tag Set

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Coordin. Conjunction</td>
<td><em>and, but, or</em></td>
<td>SYM</td>
<td>Symbol</td>
<td>+, %, &amp;</td>
</tr>
<tr>
<td>CD</td>
<td>Cardinal number</td>
<td><em>one, two, three</em></td>
<td>TO</td>
<td>“to”</td>
<td><em>to</em></td>
</tr>
<tr>
<td>DT</td>
<td>Determiner</td>
<td><em>a, the</em></td>
<td>UH</td>
<td>Interjection</td>
<td><em>ah, oops</em></td>
</tr>
<tr>
<td>EX</td>
<td>Existential “there”</td>
<td><em>there</em></td>
<td>VB</td>
<td>Verb, base form</td>
<td><em>eat</em></td>
</tr>
<tr>
<td>FW</td>
<td>Foreign word</td>
<td><em>mea culpa</em></td>
<td>VBD</td>
<td>Verb, past tense</td>
<td><em>ate</em></td>
</tr>
<tr>
<td>IN</td>
<td>Preposition/sub-conj</td>
<td><em>of, in, by</em></td>
<td>VBG</td>
<td>Verb, gerund</td>
<td><em>eating</em></td>
</tr>
<tr>
<td>JJ</td>
<td>Adjective</td>
<td><em>yellow</em></td>
<td>VBN</td>
<td>Verb, past participle</td>
<td><em>eaten</em></td>
</tr>
<tr>
<td>JJR</td>
<td>Adj., comparative</td>
<td><em>bigger</em></td>
<td>VBP</td>
<td>Verb, non-3sg pres</td>
<td><em>eat</em></td>
</tr>
<tr>
<td>JJS</td>
<td>Adj., superlative</td>
<td><em>wildest</em></td>
<td>VBZ</td>
<td>Verb, 3sg pres</td>
<td><em>eats</em></td>
</tr>
<tr>
<td>LS</td>
<td>List item marker</td>
<td><em>1, 2, One</em></td>
<td>WDT</td>
<td>Wh-determiner</td>
<td><em>which, that</em></td>
</tr>
<tr>
<td>MD</td>
<td>Modal</td>
<td><em>can, should</em></td>
<td>WP</td>
<td>Wh-pronoun</td>
<td><em>what, who</em></td>
</tr>
<tr>
<td>NN</td>
<td>Noun, sing. or mass</td>
<td><em>llama</em></td>
<td>WP$</td>
<td>Possessive wh-</td>
<td><em>whose</em></td>
</tr>
<tr>
<td>NNS</td>
<td>Noun, plural</td>
<td><em>llamas</em></td>
<td>WRB</td>
<td>Wh-adverb</td>
<td><em>how, where</em></td>
</tr>
<tr>
<td>NNP</td>
<td>Proper noun, singular</td>
<td><em>IBM</em></td>
<td>$</td>
<td>Dollar sign</td>
<td><em>$</em></td>
</tr>
<tr>
<td>NNP</td>
<td>Proper noun, plural</td>
<td><em>Carolinas</em></td>
<td>#</td>
<td>Pound sign</td>
<td><em>#</em></td>
</tr>
<tr>
<td>PDT</td>
<td>Predeterminer</td>
<td><em>all, both</em></td>
<td>“</td>
<td>Left quote</td>
<td><em>(“or “)</em></td>
</tr>
<tr>
<td>POS</td>
<td>Possessive ending</td>
<td><em>‘s</em></td>
<td>”</td>
<td>Right quote</td>
<td><em>(“ or ”)</em></td>
</tr>
<tr>
<td>PP</td>
<td>Personal pronoun</td>
<td><em>I, you, he</em></td>
<td>(</td>
<td>Left parenthesis</td>
<td><em>([, {, &lt;)</em></td>
</tr>
<tr>
<td>PPS</td>
<td>Possessive pronoun</td>
<td><em>your, one’s</em></td>
<td>)</td>
<td>Right parenthesis</td>
<td><em>(], }, &gt;)</em></td>
</tr>
<tr>
<td>RB</td>
<td>Adverb</td>
<td><em>quickly, never</em></td>
<td>;</td>
<td>Comma</td>
<td>*, ;</td>
</tr>
<tr>
<td>RBR</td>
<td>Adverb, comparative</td>
<td><em>faster</em></td>
<td>:</td>
<td>Sentence-final punct</td>
<td><em>(. ! ?)</em></td>
</tr>
<tr>
<td>RBS</td>
<td>Adverb, superlative</td>
<td><em>fastest</em></td>
<td>;</td>
<td>Mid-sentence punct</td>
<td><em>(; ... ;)</em></td>
</tr>
<tr>
<td>RP</td>
<td>Particle</td>
<td><em>up, off</em></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Disambiguation

- Resolve the ambiguities and chose the proper tag for the context
- Most English words are unambiguous (have only one tag) but many of the most common words are ambiguous
  - E.g.: “can” can be a (an auxiliary) verb or a noun
  - E.g.: statistics of Brown corpus

<table>
<thead>
<tr>
<th>Unambiguous (1 tag)</th>
<th>35,340</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ambiguous (2–7 tags)</td>
<td>4,100</td>
</tr>
<tr>
<td>2 tags</td>
<td>3,760</td>
</tr>
<tr>
<td>3 tags</td>
<td>264</td>
</tr>
<tr>
<td>4 tags</td>
<td>61</td>
</tr>
<tr>
<td>5 tags</td>
<td>12</td>
</tr>
<tr>
<td>6 tags</td>
<td>2</td>
</tr>
</tbody>
</table>
| 7 tags              | 1      | ("still")

- 11.5% word types are ambiguous
- But 40% tokens are ambiguous
(However, the probabilities of tags associated a word are not equal → many ambiguous tokens are easy to disambiguate)
Process of POS Tagging

- A String of Words
- A Specified Tagset
- Tagging Algorithm
- A Single Best Tag of Each Word

Example tags:
- Book that flight: VB DT NN .
- Does that flight serve dinner?: VBZ DT NN VB NN ?
POS Tagging Algorithms

• Fall into One of Two Classes
  • Rule-based Tagger
    – Involve a large database of hand-written disambiguation rules
      • E.g. a rule specifies that an ambiguous word is a noun rather than a verb if it follows a determiner
      • ENGTWOL: a simple rule-based tagger based on the constraint grammar architecture
  • Stochastic/Probabilistic Tagger
    – Use a training corpus to compute the probability of a given word having a given context
    – E.g.: the HMM tagger chooses the best tag for a given word (maximize the product of word likelihood and tag sequence probability)
POS Tagging Algorithms

• Transformation-based/Brill Tagger
  – A hybrid approach
  – Like rule-based approach, determine the tag of an ambiguous word based on rules
  – Like stochastic approach, the rules are automatically included from previous tagged training corpus with the machine learning technique
Rule-based POS Tagging

- **Two-stage architecture**
  - **First stage**: Use a dictionary to assign each word a list of potential part-of-speech.
  - **Second stage**: Use large lists of hand-written disambiguation rules to winnow down this list to a single part-of-speech for each word.

```
Pavlov       had shown that salivation ...
Pavlov      PAVLOV N NOM SG PROPER
had          HAVE V PAST VFIN SVO
            HAVE PCP2 SVO
shown       SHOW PCP2 SVOO SVO SV
that         ADV
            PRON DEM SG
            DET CENTRAL DEM SG
salivation   N NOM SG
```

An example for The ENGTOWL tagger

A set of 1,100 constraints can be applied to the input sentence.
Rule-based POS Tagging

- Simple lexical entries in the ENGTWOL lexicon

<table>
<thead>
<tr>
<th>Word</th>
<th>POS</th>
<th>Additional POS features</th>
</tr>
</thead>
<tbody>
<tr>
<td>smaller</td>
<td>ADJ</td>
<td>COMPARATIVE</td>
</tr>
<tr>
<td>entire</td>
<td>ADJ</td>
<td>ABSOLUTE ATTRIBUTIVE</td>
</tr>
<tr>
<td>fast</td>
<td>ADV</td>
<td>SUPERLATIVE</td>
</tr>
<tr>
<td>that</td>
<td>DET</td>
<td>CENTRAL DEMONSTRATIVE SG</td>
</tr>
<tr>
<td>all</td>
<td>DET</td>
<td>PREDETERMINER SG/PL QUANTIFIER</td>
</tr>
<tr>
<td>dog’s</td>
<td>N</td>
<td>GENITIVE SG</td>
</tr>
<tr>
<td>furniture</td>
<td>N</td>
<td>NOMINATIVE SG NOINDEFDETERMINER</td>
</tr>
<tr>
<td>one-third</td>
<td>NUM</td>
<td>SG</td>
</tr>
<tr>
<td>she</td>
<td>PRON</td>
<td>PERSONAL FEMININE NOMINATIVE SG3</td>
</tr>
<tr>
<td>show</td>
<td>V</td>
<td>IMPERATIVE VFIN</td>
</tr>
<tr>
<td>show</td>
<td>V</td>
<td>PRESENT -SG3 VFIN</td>
</tr>
<tr>
<td>show</td>
<td>N</td>
<td>NOMINATIVE SG</td>
</tr>
<tr>
<td>shown</td>
<td>PCP2</td>
<td>SVOO SVO SV</td>
</tr>
<tr>
<td>occurred</td>
<td>PCP2</td>
<td>SV</td>
</tr>
<tr>
<td>occurred</td>
<td>V</td>
<td>PAST VFIN SV</td>
</tr>
</tbody>
</table>

past participle
Rule-based POS Tagging

**ADVERBIAL-THAT RULE**

Given input: ”that”

if

(+1 A/ADV/QUANT); /* if next word is adj, adverb, or quantifier */
(+2 SENT-LIM); /* and following which is a sentence boundary, */
(NOT -1 SVOC/A); /* and the previous word is not a verb like */
/* ‘consider’ which allows adjs as object complements */
then eliminate non-ADV tags
else eliminate ADV tag

Example:

```
It isn’t **that** odd!

I consider **that** odd.
```

one

ADV

ADV

Compliment

NUM
HMM-based Tagging

- Also called Maximum Likelihood Tagging
  - Pick the most-likely tag for a word

- For a given sentence or words sequence, an HMM tagger chooses the tag sequence that maximizes the following probability

\[
\text{tag}_i = \arg \max_i P(\text{word}|\text{tag}_i) \cdot P(\text{tag}|\text{previous } n-1 \text{ tags})
\]

- N-gram HMM tagger
HMM-based Tagging

• Assumptions made here
  – Words are independent of each other
    • A word’s identity only depends on its tag
  – “Limited Horizon” and “Time Invariant” (“Stationary”)
    • A word’s tag only depends on the previous tag (limited horizon) and the dependency does not change over time (time invariance)
    • Time invariance means the tag dependency won’t change as tag sequence appears different positions of a sentence
HMM-based Tagging

• Apply bigram-HMM tagger to choose the best tag for a given word
  – Choose the tag $t_i$ for word $w_i$ that is most probable given the previous tag $t_{i-1}$ and current word $w_i$

$$t_i = \arg \max_j P(t_j | t_{i-1}, w_i)$$

– Through some simplifying Markov assumptions

$$t_i = \arg \max_j P(t_j | t_{i-1}) P(w_i | t_j)$$

Tag sequence probability  
Word/lexical likelihood
HMM-based Tagging

- Apply bigram-HMM tagger to choose the best tag for a given word

\[
t_i = \arg \max_j P(t_j | t_{i-1}, w_i)
\]

\[
= \arg \max_j P(t_j, w_i | t_{i-1}) / P(w_i | t_{i-1})
\]

\[
= \arg \max_j P(t_j, w_i | t_{i-1})
\]

\[
= \arg \max_j P(w_i, t_{i-1}, t_j)P(t_j | t_{i-1})
\]

\[
= \arg \max_j P(w_i | t_j)P(t_j | t_{i-1}) = \arg \max_j P(t_j | t_{i-1})P(w_i | t_j)
\]

The same for all tags

The probability of a word only depends on its tag
HMM-based Tagging

• Example: Choose the best tag for a given word

Secretariat/NNP is /VBZ expected/VBN to/TO race/VB tomorrow/NN

\[ P(\text{VB}|\text{TO}) \times P(\text{race}|\text{VB}) = 0.00001 \]
\[ P(\text{NN}|\text{TO}) \times P(\text{race}|\text{NN}) = 0.000007 \]

Pretend that the previous word has already tagged
HMM-based Tagging

- Apply bigram-HMM tagger to choose the best sequence of tags for a given sentence

\[
\hat{T} = \arg \max_T P(T|W) \\
= \arg \max_T \frac{P(T)P(W|T)}{P(W)} \\
= \arg \max_T P(T)P(W|T) \\
= \arg \max_{t_1, t_2, ..., t_n} P(t_1, t_2, ..., t_n)P(w_1, w_1, ..., w_n|t_1, t_2, ..., t_n) \\
= \arg \max_{t_1, t_2, ..., t_n} \prod_{i=1}^{n} \left[ P(t_i|t_1, t_2, ..., t_{i-1})P(w_i|w_1, ..., w_{i-1}, t_1, t_2, ..., t_n) \right] \\
= \arg \max_{t_1, t_2, ..., t_n} \prod_{i=1}^{n} \left[ P(t_i|t_1, t_2, ..., t_{i-1})P(w_i|t_i) \right]
\]

The probability of a word only depends on its tag.
HMM-based Tagging

- The Viterbi algorithm for the bigram-HMM tagger
HMM-based Tagging

• The Viterbi algorithm for the bigram-HMM tagger

1. Initialization \( \delta_1(k) = \pi_k P(w_i|t_k), 1 \leq k \leq J \)

2. Induction \( \delta_i(j) = \max_{1 \leq i \leq J} \delta_{i-1}(k)P(t_j|t_k)P(w_i|t_j), 2 \leq i \leq n, 1 \leq k \leq J \)

\[ \psi_i(j) = \arg\max_{1 \leq j \leq J} \delta_{i-1}(k)P(t_j|t_k) \]

3. Termination \( X_n = \arg\max_{1 \leq j \leq J} \delta_n(j) \)

for \( i := n-1 \) to 1 step -1 do

\( X_i = \psi_i(X_{i+1}) \)

end
HMM-based Tagging

• Apply trigram-HMM tagger to choose the best sequence of tags for a given sentence
  - When trigram model is used

\[
\hat{T} = \arg \max_{t_1, t_2, \ldots, t_n} \left[ P(t_1) P(t_2 | t_1) \prod_{i=3}^{n} P(t_i | t_{i-2}, t_{i-1}) \right] \left[ \prod_{i=1}^{n} P(w_i | t_i) \right]
\]

• Maximum likelihood estimation based on the relative frequencies observed in the pre-tagged training corpus (labeled data)

\[
P(t_i | t_{i-2}, t_{i-1}) = \frac{c(t_{i-2} t_{i-1} t_i)}{c(t_{i-2} t_{i-1})} \quad \text{Smoothing is needed!}
\]

\[
P(w_i | t_i) = \frac{c(w_i, t_i)}{c(t_i)}
\]
HMM-based Tagging

- Apply trigram-HMM tagger to choose the best sequence of tags for a given sentence.
HMM-based Tagging

- Probability re-estimation based on unlabeled data
  - EM (Expectation-Maximization) algorithm is applied
    - Start with a dictionary that lists which tags can be assigned to which words
      » word likelihood function can be estimated
      » tag transition probabilities set to be equal
    - EM algorithm learns (re-estimates) the word likelihood function for each tag and the tag transition probabilities
  - However, a tagger trained on hand-tagged data worked better than one trained via EM
Transformation-based Tagging

• Also called Brill tagging
  – An instance of Transformation-Based Learning (TBL)

• Spirits
  – Like the rule-based approach, TBL is based on rules that specify what tags should be assigned to what word
  – Like the stochastic approach, rules are automatically induced from the data by the machine learning technique

• Note that TBL is a supervised learning technique
  – It assumes a pre-tagged training corpus
Transformation-based Tagging

- How the TBL rules are learned
  - Three major stages
    - Label every word with its most-likely tag using a set of tagging rules
    - Examine every possible transformation (rewrite rule), and select the one that results in the most improved tagging (supervised!)
    - Re-tag the data according this rule
  - The above three stages are repeated until some stopping criterion is reached
    - Such as insufficient improvement over the previous pass
Transformation-based Tagging

• Example

\[ P(\text{NN}|\text{race}) = 0.98 \]
\[ P(\text{VB}|\text{race}) = 0.02 \]

So, race will be initially coded as NN

(label every word with its most-likely tag)

1. is/\text{VBZ} expected/\text{VBN} to/\text{To} race/\text{NN} tomorrow/\text{NN}

2. the/\text{DT} race/\text{NN} for/\text{IN} outer/\text{JJ} space/\text{NN}

Refer to the correct tag
Information of each word, and find the tag of race in “1” is wrong

Learn/pick a most suitable transformation rule: (by examining every possible transformation)

Change NN to VB while the previous tag is TO

Rewrite rule: expected/\text{VBN} to/\text{To} race/\text{NN} \rightarrow \text{expected/\text{VBN}} to/\text{To} race/\text{VB}
Transformation-based Tagging

• Templates (abstracted transforms)
  – The set of possible transformation may be infinite
  • Should limit the set of transformations
  • The design of a small set of templates is needed

<table>
<thead>
<tr>
<th>#</th>
<th>Change tags</th>
<th>From</th>
<th>To</th>
<th>Condition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NN</td>
<td>VB</td>
<td></td>
<td>Previous tag is TO</td>
<td>to/TO race/NN → VB</td>
</tr>
<tr>
<td>2</td>
<td>VBP</td>
<td>VB</td>
<td></td>
<td>One of the previous 3 tags is MD</td>
<td>might/MD vanish/VBP → VB</td>
</tr>
<tr>
<td>3</td>
<td>NN</td>
<td>VB</td>
<td></td>
<td>One of the previous 2 tags is MD</td>
<td>might/MD not reply/NN → VB</td>
</tr>
<tr>
<td>4</td>
<td>VB</td>
<td>NN</td>
<td></td>
<td>One of the previous 2 tags is DT</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>VBD</td>
<td>VBN</td>
<td></td>
<td>One of the previous 3 tags is VBN</td>
<td></td>
</tr>
</tbody>
</table>
Transformation-based Tagging

- Templates (abstracted transforms)

```
<table>
<thead>
<tr>
<th>Schema</th>
<th>t_{i-3}</th>
<th>t_{i-2}</th>
<th>t_{i-1}</th>
<th>t_i</th>
<th>t_{i+1}</th>
<th>t_{i+1}</th>
<th>t_{i+3}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**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 n’t</td>
</tr>
</tbody>
</table>

**Table 10.8** Examples of some transformations learned in transformation-based tagging.
Transformation-based Tagging

- Algorithm

```plaintext
function TBL(corpus) returns transforms-queue
  INITIALIZE-WITH-MOST-LIKELY-TAGS(corpus)
  until end condition is met do
    templates ← GENERATE-POTENTIAL-RELEVANT-TEMPLATES
    best-transform ← GET-BEST-TRANSFORM(corpus, templates)
    APPLY-TRANSFORM(best-transform, corpus)
    ENQUEUE(best-transform-rule, transforms-queue)
  end
  return(transforms-queue)

function GET-BEST-TRANSFORM(corpus, templates) returns transform
  for each template in templates
    (instance, score) ← GET-BEST-INSTANCE(corpus, template)
    if (score > best-transform.score) then best-transform ← (instance, score)
  end
  return(best-transform)

function GET-BEST-INSTANCE(corpus, template) returns transform
  for from-tag ← from-tag→tag→1 to tag→n do
    for to-tag ← from-tag→1 to tag→n do
      for pos ← from 1 to corpus-size do
        if (correct-tag(pos) = to-tag & current-tag(pos) = from-tag)
          num-good-transforms(current-tag(pos-1))++
        elseif (correct-tag(pos) = from-tag & current-tag(pos) = from-tag)
          num-bad-transforms(current-tag(pos-1))++
        end
        best-Z ← ARGMAX((num-good-transforms(pos) - num-bad-transforms(pos))
          if (num-good-transforms(best-Z) > best-instance.Z) then
            best-instance ← “Change tag from from-tag to to-tag
            if previous tag is best-Z”
          end
      end
    end
  end
  return(best-instance)

procedure APPLY-TRANSFORM(transform, corpus)
  for pos ← from 1 to corpus-size do
    if (current-tag(pos) = best-rule-from)
      & (current-tag(pos-1) = best-rule-prev)
    current-tag(pos) ← best-rule-to
  end
```

The `GET_BEST_INSTANCE` procedure in the example algorithm is "Change tag from X to Y if the previous tag is Z".
Multiple Tags and Multi-part Words

• Multiple tags
  – A word is ambiguous between multiple tags and it is impossible or very difficult to disambiguate, so multiple tags is allowed, e.g.
    • adjective versus preterite versus past participle (JJ/VBD/VBN)
    • adjective versus noun as prenominal modifier (JJ/NN)

• Multi-part words
  – Certain words are split or some adjacent words are treated as a single word
    would/MD n’t/RB Children/NNS ‘s/POS
    in terms of (in/Ii31 terms/Ii32 of/Ii33)
Tagging of Unknown Words

• Simplest unknown-word algorithm
  – Pretend that each unknown word is ambiguous among all possible tags, with equal probability
  – Must rely solely on the contextual POS-trigram to suggest the proper tag

• Slightly more complex algorithm
  – Based on the idea that the probability distribution of tags over unknown words is very similar to the distribution of tags over words that occurred only once in a training set
  – The likelihood for an unknown word is determined by the average of the distribution over all singleton in the training set (similar to Good-Turing?)
Tagging of Unknown Words

• Most-powerful unknown-word algorithm
  – Hand-designed features
    • The information about how the word is spelled (inflectional and derivational features), e.g.:
      – Words end with s (→ plural nouns)
      – Words end with ed (→ past participles)
    • The information of word capitalization (initial or non-initial) and hyphenation
      \[ P(w|t_i) = p(\text{unknown-word}|t_i) \cdot p(\text{capital}|t_i) \cdot p(\text{endings/hyph}|t_i) \]
  – Features induced by machine learning
    • E.g.: TBL algorithm uses templates to induce useful English inflectional and derivational features and hyphenation
Evaluation of Taggers

• Compare the tagged results with a human labeled Gold Standard test set in percentages of correction
  – Most tagging algorithms have an accuracy of around 96~97% for the sample tagsets like the Penn Treebank set
  – Upper bound (ceiling) and lower bound (baseline)
    • Ceiling: is achieved by seeing how well humans do on the task
      – A 3~4% margin of error
    • Baseline: is achieved by using the unigram most-like tags for each word
      – 90~91% accuracy can be attained
Error Analysis

• Confusion matrix

<table>
<thead>
<tr>
<th></th>
<th>IN</th>
<th>JJ</th>
<th>NN</th>
<th>NNP</th>
<th>RB</th>
<th>VBD</th>
<th>VBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>-</td>
<td>-.2</td>
<td>3.3</td>
<td>2.1</td>
<td>.7</td>
<td>.2</td>
<td>2.7</td>
</tr>
<tr>
<td>JJ</td>
<td>.2</td>
<td>-</td>
<td>.8</td>
<td>.2</td>
<td>.7</td>
<td>.2</td>
<td>.2</td>
</tr>
<tr>
<td>NN</td>
<td>8.7</td>
<td>-</td>
<td>4.1</td>
<td>2.1</td>
<td>1.7</td>
<td>.2</td>
<td>.2</td>
</tr>
<tr>
<td>NNP</td>
<td>.2</td>
<td>3.3</td>
<td>-</td>
<td>.2</td>
<td>.7</td>
<td>.2</td>
<td>.2</td>
</tr>
<tr>
<td>RB</td>
<td>2.2</td>
<td>2.0</td>
<td>.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4.4</td>
</tr>
<tr>
<td>VBD</td>
<td>.3</td>
<td>.5</td>
<td>-</td>
<td>-</td>
<td>2.6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VBN</td>
<td>2.8</td>
<td>-</td>
<td>.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

• Major problems facing current taggers
  – NN (noun) versus NNP (proper noun) and JJ (adjective)
  – RP (particle) versus RB (adverb) versus JJ
  – VBD (past tense verb) versus VBN (past participle verb) versus JJ
Applications of POS Tagging

• Tell what words are likely to occur in a word’s vicinity
  – E.g. the vicinity of the possessive or person pronouns

• Tell the pronunciation of a word
  – DIScount (noun) and disCOUNT (verb) …

• Advanced ASR language models
  – Word-class N-grams

• Partial parsing
  – A simplest one: find the noun phrases (names) or other phrases in a sentence
Applications of POS Tagging

• Information retrieval
  – Word stemming
  – Help select out nouns or important words from a doc
  – Phrase-level information

  United, States, of, America → “United States of America”
  secondary, education → “secondary education”

• Phrase normalization

  Book publishing, publishing of books

• Information extraction
  – Semantic tags or categories
Applications of POS Tagging

• Question Answering
  – Answer a user query that is formulated in the form of a question by return an appropriate noun phrase such as a location, a person, or a date
  • E.g. “Who killed President Kennedy?”

In summary, the role of taggers appears to be a fast lightweight component that gives sufficient information for many applications
  – But not always a desirable preprocessing stage for all applications
  – Many probabilistic parsers are now good enough!
Class-based N-grams

- Use the lexical tag/category/class information to augment the $N$-gram models

$$P(w_n | w_{n-N+1}^{n-1}) = P(w_n | c_n)P(c_n | c_{n-N+1}^{n-1})$$

- Maximum likelihood estimation

$$P(w_i | c_j) = \frac{C(w)}{C(c)}$$

$$P(c_j | c_k) = \frac{C(c_k c_j)}{\sum_l C(c_l c_j)}$$

Constraints: a word may only belong to one lexical category
行政院院長決定廢核四