Part-of-Speech Tagging

Torbjörn Lager
Department of Linguistics
Göteborg University

Part-of-Speech Tagging: Definition

- From Jurafsky & Martin 2000:
  - Part-of-speech tagging is the process of assigning a part-of-speech or other lexical class marker to each word in a corpus.
  - The input to a tagging algorithm is a string of words and a specified tagset. The output is a single best tag for each word.
  - A bit too narrow for my taste...

Part-of-Speech Tagging: Example 1

- Input:
  He can can a can

- Output:
  He/pron can/aux can/vb a/det can/n
  Another possible output:
  He/{pron} can/{aux,n} can/{vb} a/{det} can/{n,vb}

Tag Sets

- The Penn Treebank tag set
  (see HTML page)

Why Part-of-Speech Tagging?

- A first step towards parsing
- A first step towards word sense disambiguation
- Provide clues to pronunciation
  - "object" -> OBJECT or object
  - (but note: BAan vs baNAN)
- Research in Corpus Linguistics

Part-of-Speech Tagging: Example 2

- I can light a fire and you can open a can of beans. Now the can is open and we can eat in the light of the fire
Relevant Information

- Lexical information
- Local contextual information

Part-of-Speech Tagging: Example 2

- I can light a fire and you can open a can of beans. Now the can is open and we can eat in the light of the fire.
- I/PRP can/MD light/VB a/DT fire/NN and/CC you/PRP can/MD open/VB a/DT can/NN of/IN beans/NNS /. Now/RB the/DT con/NN is/VBZ open/JJ and/CC we/PRP can/MD eat/VB in/IN the/DT light/NN of/IN the/DT fire/NN ./.

Part-of-Speech Tagging

Knowledge

Text

Processor

POS Tagged Text

Needed:
- Some strategy for representing the knowledge
- Some method for acquiring the knowledge
- Some method of applying the knowledge

Approaches to PoS Tagging

- The bold approach: 'Use all the information you have and guess'
- The whimsical approach: 'Guess first, then change your mind if necessary!
- The cautious approach: 'Don’t guess, just eliminate the impossible!'

Some POS-Tagging Issues

- Accuracy
- Speed
- Space requirements
- Learning
- Intelligibility

Cutting the Cake

- Tagging methods
  - Rule based
  - Statistical
  - Mixed
  - Other methods
- Learning methods
  - Supervised learning
  - Unsupervised learning
HMM Tagging

- The bold approach: 'Use all the information you have and guess''
- Statistical method
- Supervised (or unsupervised) learning

The Naive Approach and its Problem

- Traverse all the paths compatible with the input and then pick the most probable one
- Problem:
  - There are 27 paths in the HMM for S="he can can a can"
  - Doubling the length of S (with a conjunction in between) → 729 paths
  - Doubling S again → 531431 paths!
  - Exponential time complexity!

Solution

- Use the Viterbi algorithm
- Tagging can be done in time proportional to the length of input.
- How and Why does the Viterbi algorithm work? We save this for later...

Training an HMM

- Estimate probabilities from relative frequencies.
  - Output probabilities P(w|t): the number of occurrences of w tagged as t, divided by the number of occurrences of t.
  - Transitional probabilities P(t1|t2): the number of occurrences of t1 followed by t2, divided by the number of occurrences of t2.
- Use smoothing to overcome the sparse data problem (unknown words, uncommon words, uncommon contexts)

Transformation-Based Learning

- The whimsical approach: 'Guess first, then change your mind if necessary'
- Rule based tagging, statistical learning
- Supervised learning
Transformation-Based Tagging:
Small Part-of-Speech Tagging Example

rules

pos:NN>VB <- pos:TO@[ -1] o
pos:VB>NN <- pos:DT@[ -1] o

Input
She decided to table her data

lexicon

data:NN
decided:VB
her:PN

table:NN VB
to:TO

A Longer Rule Sequence

tag:'NN'>>'VB' <- tag:'TO'@[ -1] o
tag:'VB'>>'VB' <- tag:'MD'@[ -1,-2,-3] o
tag:'NN'>>'VB' <- tag:'MD'@[ -1,-2] o
tag:'VB'>>'NN' <- tag:'DT'@[ -1,-2] o
tag:'VB'>>'VB' <- tag:'VBZ'@[ -1,-2,-3] o
tag:'VB'>>'VB' <- tag:'PRP'@[ -1] o
tag:'POS'>>'VBZ' <- tag:'PRP'@[ -1] o
tag:'VB'>>'VBD' <- tag:'VBN'@[ -1] o
tag:'VBD'>>'VBZ' <- tag:'VBN'@[ -1] o

tag:'IN'>>'IN' <- wda:as@[0] & wda:as@[2] o
tag:'IN'>>'VBD' <- tag:'VBN'@[1,2] o
tag:'VB'>>'VBD' <- tag:'PRP'@[ -1] o
tag:'IN'>>'VBD' <- tag:'VBZ'@[0] o

...
Transformation-Based Painting

Learning Transformation Rules

Various Corpora

- Training corpus
  w0 w1 w2 w3 w4 w5 w6 w7 w8 w9 w10
- Current corpus (CC 1)
  dt vb nn dt vb kn dt vb ab dt vb
- Reference corpus
  dt nn vb dt nn kn dt jj kn dt nn

Stop when score of best rule falls below threshold.
Learning Transformation Rules

- **Training Corpus**
- **Baseline System**
- **Current Corpus**

Derive and Score Candidate Rules
- **Rule Templates**
- **Reference Corpus**
- Select Best Rule
- Apply Rule

Learned Rule Sequence

Rule Templates

- In TBL, only rules that are instances of **templates** can be learned.
- For example, the rules
  
  `tag:VB>'NN' ⇐ tag:'DT'@[-1].`  
  `tag:'NN'>'VB' ⇐ tag:'DT'@[-1].`

- are instances of the template
  
  `tag:A>B ⇐ tag:C@[-1].`

Score, Accuracy and Thresholds

- The score of a rule is the number of its positive matches minus the number of its negative instances:
  
  $$\text{score}(R) = |\text{pos}(R)| - |\text{neg}(R)|$$

- The accuracy of a rule is its number of positive matches divided by the total number of matches of the rule:
  
  $$\text{accuracy}(R) = \frac{|\text{pos}(R)|}{|\text{pos}(R)| + |\text{neg}(R)|}$$

- The score threshold and the accuracy threshold are the lowest score and the lowest accuracy, respectively, that the highest scoring rule must have in order to be considered.
- In ordinary TBL, we work with an accuracy threshold < 0.5.

Derive and Score Candidate Rule 1

- **Template** = `tag:_>_ ⇐ tag:_@[-1]`
- **Rule** = `tag:vb>nn ⇐ tag:dt@[-1]`

<table>
<thead>
<tr>
<th>Ref.</th>
<th>vb</th>
<th>vb</th>
<th>vb</th>
<th>vb</th>
<th>vb</th>
<th>vb</th>
<th>vb</th>
<th>vb</th>
</tr>
</thead>
<tbody>
<tr>
<td>cc</td>
<td>cc</td>
<td>cc</td>
<td>cc</td>
<td>cc</td>
<td>cc</td>
<td>cc</td>
<td>cc</td>
<td>cc</td>
</tr>
</tbody>
</table>

- **pos(R1) = 3**
- **neg(R1) = 1**
- **score(R1) = pos(R1) - neg(R1) = 3 - 1 = 2**

Derive and Score Candidate Rule 2

- **Template** = `tag:_>_ ⇐ tag:_@[-1]`
- **Rule** = `tag:nn>vb ⇐ tag:vb@[-1]`

<table>
<thead>
<tr>
<th>Ref.</th>
<th>vb</th>
<th>vb</th>
<th>vb</th>
<th>vb</th>
<th>vb</th>
<th>vb</th>
<th>vb</th>
<th>vb</th>
</tr>
</thead>
<tbody>
<tr>
<td>cc</td>
<td>cc</td>
<td>cc</td>
<td>cc</td>
<td>cc</td>
<td>cc</td>
<td>cc</td>
<td>cc</td>
<td>cc</td>
</tr>
</tbody>
</table>

- **pos(R2) = 1**
- **neg(R2) = 0**
- **score(R2) = pos(R2) - neg(R2) = 1 - 0 = 1**
Learning Transformation Rules

- Training Corpus
- Derive and Score Candidate Rules
- Rule Templates
- Reference Corpus
- Select Best Rule
- Current Corpus
- Apply Rule
- Learned Rule Sequence
- Stop when score of best rule falls below threshold.

Select Best Rule

- Current ranking of rule candidates
  - R1 = tag:vb>nn <- tag:dt@[-1] Score = 2
  - R2 = tag:nn>vb <- tag:vb@[-1] Score = 1
  ...
- If score threshold <= 2 then select R1, else if score threshold > 2, terminate.

Constraint-Grammar Tagging

- Due to Fred Karlsson et al.
- The cautious approach: ‘Don’t guess, just eliminate the impossible!’
- Rule based
- No learning (‘learning by injection’)

Constraint Grammar Example

- I can light a fire and you can open a can of beans. Now the can is open and we can eat in the light of the fire.
- /I(PR) can/(MD NN) light/(JJ NN,VB) a/(DT) fire/(NN)
  and/(CC) you/(PRP) can/(MD NN) open/(JJ VB) a/(DT)
  can/(MD NN) of/(IN) beans/(NNS) ./(). Now/(RB) the/(DT)
  can/(MD NN) is/(VBZ) open/(JJ VB) and/(CC) we/(PRP)
  can/(MD NN) eat/(VB) in/(IN) the/(DT) light/(JJ NN,VB)
  of/(IN) the/(DT) fire/(NN) ./()
Evaluation

- Two reasons for evaluating:
  - Compare with other people's methods/systems
  - Compare with earlier versions of your own system
- Accuracy (recall and precision)
- Baseline
- Ceiling
- N-fold cross-validation methodology => Good use of the data + More statistically reliable results.

Assessing the Taggers

- Accuracy
- Speed
- Space requirements
- Learning
- Intelligibility

Demo Taggers

- Transformation-Based Tagger:
  - [www.ling.gu.se/~lager/Home/brilltagger_ui.html](http://www.ling.gu.se/~lager/Home/brilltagger_ui.html)
- Constraint-Grammar Tagger
  - [www.ling.gu.se/~lager/Home/cptagger_ui.html](http://www.ling.gu.se/~lager/Home/cptagger_ui.html)
- Featuring tracing facilities!
- Try it yourself!