Brill Tagging

Reading: Chap 8, Jurafsky & Martin

Faculty: B. Senthil Kumar

Natural Language Processing

tagging

Automatic approaches 1: rule-based tagging

Automatic approaches 2: stochastic tagging

Automatic approaches 3: transformation-based tagging

Other issues: tagging unknown words, evaluation

Transformation-Based Tagging

- An instance of Transformation-Based Learning (TBL)
- Combination of Rule-based and stochastic tagging methodologies
  - Like rule-based taggers, TBL is based on rules that specify what tags should be assigned to what words
  - Like stochastic taggers, TBL is a machine learning technique, in which rules are automatically induced from the data
- Input:
  - tagged corpus

Transformation-Based Tagging (cont.)

- Basic Idea:
  - Set the most probable tag for each word as a start value
  - Change tags according to rules of type “if word-1 is a determiner and word is a verb then change the tag to noun” in a specific order
- Training is done on tagged corpus:
  - Write a set of rule templates
  - Among the set of rules, find one with highest score
  - Continue from 2 until lowest score threshold is passed
  - Keep the ordered set of rules
- Rules make errors that are corrected by later rules

TBL Rule Application

Tagger labels every word with its most-likely tag
(tagged corpus)

For example: *race* has the following probabilities in the

Brown corpus:

P(NN|race) = .98

P(VB|race)= .02

… is/VBZ expected/VBN to/TO race/NN tomorrow/NN

… the/DT race/NN for/IN outer/JJ space/NN

TBL Rule Application

After selecting the most-likely tag, Brill's tagger applies its transformation rules

Transformation rules make changes to tags:

“Change NN to VB when previous tag is TO”

… is/VBZ expected/VBN to/TO race/NN tomorrow/NN

becomes

… is/VBZ expected/VBN to/TO race/VB tomorrow/NN

TBL: The Algorithm

- Step 1: Label every word with *most-likely tag*
- Step 2: Check every possible transformation & select one that results in the most improves tagging
- Step 3: *Re-tag* corpus applying the rules
- Repeat 2-3 until some stopping criterion is reached, e.g., X % correct with respect to training corpus
- RESULT: Sequence of transformation rules

TBL: Rule Learning (cont'd)

- Problem: Could apply transformations ad infinitum!
- Constrain the set of transformations with "templates":
  - Replace tag X with tag Y, provided tag Z or word Z' appears in some position
- Rules are learned in ordered sequence
- Rules may interact.
- Rules are compact and can be inspected by humans

TBL: Rule Learning (cont'd)

- GET_BEST_TRASFORMATION & GET_BEST_INSTANCE are the two important functions in TBL algorithm for rule learning
- GET_BEST_TRASFORMATION is called with a list of potential templates; for each template, it calls GET_BEST_INSTANCE
- GET_BEST_INSTANCE iteratively tests every possible instantiation of each template by filling in specific values for the tag variables **a**, **b**, **z**, and **w**.

Templates for TBL

Brill's templates. Each begins with *"Change tag* **a** *to tag* **b** *when:..."*

The preceding (following) word is tagged **z**.
The word two before (after) is tagged **z**.
One of the two preceding (following) words is tagged **z**.
One of the three preceding (following) words is tagged **z**.
The preceding word is tagged **z** and the following word is tagged **w**.
The preceding (following) word is tagged **z** and the word two before (after) is tagged **w**.

rules learned by Brill's original tagger

#	Change tags From	To	Condition	Example
1	NN	VB	Previous tag is TO	to/TO race/NN → VB
2	VBP	VB	One of the previous 3 tags is MD	might/MD vanish/VBP → VB
3	NN	VB	One of the previous 2 tags is MD	might/MD not reply/NN → VB
4	VB	NN	One of the previous 2 tags is DT	
5	VBD	VBN	One of the previous 3 tags is VBZ	

TBL: Problems

- Execution Speed: TBL tagger is slower than HMM approach
  - Solution: compile the rules to a Finite State Transducer (FST), Roche and Schabes (1997)

Outline

Automatic approaches 1: rule-based tagging

Automatic approaches 2: stochastic tagging

Automatic approaches 3: transformation-based tagging

Other issues: multiple tags, tagging unknown words

Multiple Tags and Multiple Words

- Tag indeterminacy occurs when a word is ambiguous between multiple tags
- Penn Treebank and BNC allow the use of multiple tags
- Ex: adjective vs. preterite vs. past participle (JJ/VBD/VBN)
- Three ways to deal tag indeterminacy:
    - Replace the indeterminate tags with only one tag
    - In testing, count a tagger as having correctly tagged an intermediate token if it gives either of correct tags. In training choose only one of the tags for the word
    - Treat indeterminate tag as single complex tag

Multiple Tags and Multiple Words

- Second issue: multi-part words
- Treebank tagset:

  a New York City firm (tagged as five separate words)

  a/DT New/NNP York/NNP City/NNP firm/NN
- C5 and C7 tagsets allow prepositions like "*in terms of*" to be treated as single word by adding numbers to each tag: in/II31 terms/II32 of/II33

Tagging Unknown Words

- Proper names and acronyms are created often
- New common nouns and verbs enter the language at a high rate
- Need some method for guessing the tag of unknown word
- *Method 1*: assume they are nouns
- *Method 2*: assume the unknown words have a probability distribution similar to words only occurring once (*hapax legomena*) in the training set

Tagging Unknown Words

- *Method 3*: Use morphological information,
- words ending with *–ed* tend to be tagged VBN – past participles.
- Words end in the letter -s are plural nouns (NNS)
- Words start with capital letters are likely to be proper nouns (NP)
- Hyphenated words are most likely to be adjectives (JJ)

Tagging Unknown Words

- Weischedel et al. (1993) – four kinds of orthographic features:
  - 3 inflectional endings (-ed, -s, -ing)
  - 32 derivational endings (-ion, -al, -ive, -ly)
  - 4 values of capitalization (word is sentence-initial)
  - hyphenation
- Used the following to compute the likelihood of an unknown word:

  P(Wi|ti) = p(unknown-word|ti) * p(capital|ti) * p(endings/hyph|ti)

Thank You

References:

Speech and Language Processing, Jurafsky & Martin