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

| Tag | Part Of Speech |
|---|---|
| AT | article |
| BEZ | the word *is* |
| IN | preposition |
| JJ | adjective |
| JJR | comparative adjective |
| MD | modal |
| NN | singular or mass noun |
| NNP | singular proper noun |
| NNS | plural noun |
| PERIOD | . : ? ! |
| PN | personal pronoun |
| RB | adverb |
| RBR | comparative adverb |
| TO | the word *to* |
| VB | verb, base form |
| VBD | verb, past tense |
| VBG | verb, present participle, gerund |
| VBN | verb, past participle |
| VBP | verb, non-3rd person singular present |
| VBZ | verb, 3rd singular present |
| WDT | *wh*- determiner (*what*, *which*) |

**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)

| | |
|---|---|
| $w_i$ | the word at position $i$ in the corpus |
| $t_i$ | the tag of $w_i$ |
| $w_{i,i+m}$ | the words occurring at positions $i$ through $i + m$ (alternative notations: $w_i \cdots w_{i+m}$, $w_i, \ldots, w_{i+m}$, $w_{i(i+m)}$) |
| $t_{i,i+m}$ | the tags $t_i \cdots t_{i+m}$ for $w_i \cdots w_{i+m}$ |
| $w^l$ | the $l^{\text{th}}$ word in the lexicon |
| $t^j$ | the $j^{\text{th}}$ tag in the tag set |
| $C(w^l)$ | the number of occurrences of $w^l$ in the training set |
| $C(t^j)$ | the number of occurrences of $t^j$ in the training set |
| $C(t^j, t^k)$ | the number of occurrences of $t^j$ followed by $t^k$ |
| $C(w^l : t^j)$ | the number of occurrences of $w^l$ that are tagged as $t^j$ |
| $T$ | number of tags in tag set |
| $W$ | number of words in the lexicon |
| $n$ | sentence length |

**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

| First tag | Second tag AT | BEZ | IN | NN | VB | PERIOD |
|---|---|---|---|---|---|---|
| AT | 0 | 0 | 0 | 48636 | 0 | 19 |
| BEZ | 1973 | 0 | 426 | 187 | 0 | 38 |
| IN | 43322 | 0 | 1325 | 17314 | 0 | 185 |
| NN | 1067 | 3720 | 42470 | 11773 | 614 | 21392 |
| VB | 6072 | 42 | 4758 | 1476 | 129 | 1522 |
| PERIOD | 8016 | 75 | 4656 | 1329 | 954 | 0 |

**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



| | AT | BEZ | IN | NN | VB | PERIOD |
|---|---|---|---|---|---|---|
| *bear* | 0 | 0 | 10 | 0 | 43 | 0 |
| *is* | 0 | 10065 | 0 | 0 | 0 | 0 |
| *move* | 0 | 0 | 0 | 36 | 133 | 0 |
| *on* | 0 | 0 | 5484 | 0 | 0 | 0 |
| *president* | 0 | 0 | 0 | 382 | 0 | 0 |
| *progress* | 0 | 0 | 0 | 108 | 4 | 0 |
| *the* | 69016 | 0 | 0 | 0 | 0 | 0 |
| . | 0 | 0 | 0 | 0 | 0 | 48809 |

**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

| Feature | Value | NNP | NN | NNS | VBG | VBZ |
|---|---|---|---|---|---|---|
| unknown word | yes | 0.05 | 0.02 | 0.02 | 0.005 | 0.005 |
| | no | 0.95 | 0.98 | 0.98 | 0.995 | 0.995 |
| capitalized | yes | 0.95 | 0.10 | 0.10 | 0.005 | 0.005 |
| | no | 0.05 | 0.90 | 0.90 | 0.995 | 0.995 |
| ending | *-s* | 0.05 | 0.01 | 0.98 | 0.00 | 0.99 |
| | *-ing* | 0.01 | 0.01 | 0.00 | 1.00 | 0.00 |
| | *-tion* | 0.05 | 0.10 | 0.00 | 0.00 | 0.00 |
| | other | 0.89 | 0.88 | 0.02 | 0.00 | 0.01 |

**Table 10.5** Table of probabilities for dealing with unknown words in tagging. For example, $P$(unknown word = yes|NNP) = 0.05 and $P$(ending = *-ing*|VBG) = 1.0.

$$P(w^l|t^j) = \frac{1}{Z} P(unknown|t^j) \times P(capitalized|t^j) \times P(endings|t^j)$$

# 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 emmision) probabilities

$$b_{jl} = \frac{b^*_{jl} C(w^l)}{\sum_{w^m} b^*_{jm} C(w^m)}$$

$$b^*_{jl} = \begin{cases} 0 \text{ if } t^j \text{ is not a part of speech for } w^l \\ 1/T(w^l) \quad \text{otherwise} \end{cases}$$

Assume $b^*_{jl} = P(t^j \mid w^l) = 1 / T(w^l)$, i.e. uniform

We want $P(w^l \mid t^j) = \dfrac{P(t^j \mid w^l)P(w^l)}{P(t^j)}$

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

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

$$= \frac{\dfrac{C(w^l)}{T(w^l)}}{\sum_{w^m} \dfrac{C(w^m)}{T(w^m)}}$$

# 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_I
- Which transformations / corrections admissible ?
- How to learn these ?

# Table 10.7/10.8

| Schema | $t_{i-3}$ | $t_{i-2}$ | $t_{i-1}$ | $t_i$ | $t_{i+1}$ | $t_{i+1}$ | $t_{i+3}$ |
|---|---|---|---|---|---|---|---|
| 1 | | | | * | | | |
| 2 | | | | * | | | |
| 3 | | | | * | | | |
| 4 | | | | * | | | |
| 5 | | | | * | | | |
| 6 | | | | * | | | |
| 7 | | | | * | | | |
| 8 | | | | * | | | |
| 9 | | | | * | | | |

**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."

| Source tag | Target tag | Triggering environment |
|---|---|---|
| NN | VB | previous tag is TO |
| VBP | VB | one of the previous three tags is MD |
| JJR | RBR | next tag is JJ |
| VBP | VB | one of the previous two words is *n't* |

**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
3 **for** $k := 0$ **step** 1 **do**
4 $\quad \nu :=$ the transformation $u_i$ that minimizes $E(u_i(C_k))$
6 $\quad$ **if** $(E(C_k) - E(\nu(C_k))) < \epsilon$ **then break fi**
7 $\quad C_{k+1} := \nu(C_k)$
8 $\quad \tau_{k+1} := \nu$
9 **end**
10 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| r 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 -> S) = Count(N -> S) / Count(N)

# Conclusions

- Pos-tagging as an application of SNLP
- VMM, HMMs, TBL
- Statistical tagggers
  - 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.