POS tagging

LING 570

Fei Xia

Outline

- The POS tagging task

- Rule-based approach

- Statistical approach
  - N-gram model: HMM
  - MaxEnt model: later in ling570
  - Other models: in Ling572 and beyond

The task

- Training data: a tagged corpus
- Build a system:
  - Input: w_1 w_2 …. w_n
  - Output: w_1/t_1 w_2/t_2 …. w_n/t_n
- POS tags:
  - Open class: noun, verb, adj, adv
  - Closed class: prep, det, pron, conj, particles, …
- Tagsets:
  - 30 tags or more is pretty common.
  - Ex: how many tags for verbs?

Why POS tagging?

- As a preprocessing step for parsing, chunking, etc.
  - Chunking: /Det? Adj* N* N/
  - Parsing: VP ➔ V NP vs. VP ➔ buy NP

- Text-to-speech: Please record the lecture

- Morphological analysis:
  - Ex: saw ➔ see +V +past

    saw ➔ saw +N + PL

Main problem: ambiguity

- Example: book a flight; buy a book

- How hard is the tagging problem?
  - Many frequent words are ambiguous.
  - Penn English Treebank (PTB):
    - Unigram: 91%
    - Trigram: 93%
    - Best result: 97-98%
    - Upper bound: 97-98% (?)
  - The tagging problem may be harder for
    - other domains
    - other languages

Main approaches

- Rule-based approach:

- Stochastic approach: Choose t_1 t_2 …. t_n that maximizes $P(t_1^n | w_1^n)$
  - N-gram models:

  - Use a classifier with beam search
    - Ex: Decision Tree, MaxEnt, Boosting, SVM, …

  - Use sequence labeling algorithms
    - Ex: HMM, CRF, TBL

➔ Most of the algorithms will be covered in LING 572.

➔ Today we will focus on N-gram models.

Evaluation

- Train your model on the training data
- Test on unseen test data to obtain the best tag sequence.
- Accuracy: the percentage of words in the test data that are correctly tagged:
  - System: John/N called/V this/PN number/N
  - Gold: John/N called/V this/DT number/N
  - Accuracy is 3/4

Rule-based approach

POS tagger for English

- Human knowledge

- Annotated data:
  - John/NNP will/MD book/VB the/DT flight/NN tomorrow/NN
  - Mary/NNP bought/VBD a/DT book/NN

- Rules:
  - NN => VB if the word follows a MD

- Transformation-based learning (TBL)

N-gram tagger

Building a statistical system

- Collect data and divide it into training, development, and testing or use n-fold cross validation
- Modeling:
  - What is the function to optimize? e.g., $P(y \mid x)$, $P(x, y)$
  - How to decompose it to something that can be estimated?
- Training: estimate the parameters from the training data
- Decoding: run the model on the test data
- Evaluation: compare the system output with the gold standard

Notation

$$w_1^n\!: \; w_1 \; w_2 \; \dots \; w_n$$

$$t_1^n\!: \; t_1 \; t_2 \; \dots \; t_n$$

$$max_y P(y|x)$$

$$y^* = arg\ max_y P(y|x)$$

N-gram POS tagger: modeling

$$arg\,max_{t_1^n} P(t_1^n|w_1^n)$$

$$= arg\,max_{t_1^n} \frac{P(t_1^n) * P(w_1^n|t_1^n)}{P(w_1^n)}$$

$$= arg\,max_{t_1^n} P(t_1^n) * P(w_1^n|t_1^n)$$

$$P(t_1^n) \approx \prod_i P(t_i|t_{i-N+1}^{i-1})$$

$$P(w_1^n|t_1^n) = \prod_i P(w_i|t_1^n, w_1^{i-1}) \approx \prod_i P(w_i|t_i)$$

N-gram POS tagger (cont)

$$argmax_{t_1^n} P(t_1^n | w_1^n)$$

$$\approx argmax_{t_1^n} \prod_i P(w_i | t_i) P(t_i | t_{i-N+1}^{i-1})$$

Bigram model: $$\prod_i P(w_i | t_i) P(t_i | t_{i-1})$$

Trigram model: $$\prod_i P(w_i | t_i) P(t_i | t_{i-2}, t_{i-1})$$

Bigram model: training

$$\prod_i P(w_i|t_i)P(t_i|t_{i-1})$$

Training: How to estimate $P(w_i \mid t_i)$ and $P(t_i \mid t_{i-1})$?

- Supervised learning (tags in the training data are known): ML estimation

- Unsupervised learning (tags in the training data are unknown): forward-backward algorithm

Bigram training: ML estimation

$$P(w_i|t_i) = \frac{Cnt(w_i, t_i)}{Cnt(t_i)}$$

$$P(t_i|t_{i-1}) = \frac{Cnt(t_{i-1}, t_i)}{Cnt(t_{i-1})}$$

Bigram model: decoding

- Given $P(w_i \mid t_i)$ and $P(t_i \mid t_{i-1})$, how to find the best tag sequence for a sentence?

➔Use Viterbi algorithm for HMM

- The task of determining which sequence of variables is the underlying source of observations is called the **decoding** task.

Coming next

- Hidden Markov Model (HMM)
- Use a classifier