A Practical Part-of-Speech Tagger

Doug Cutting
Julian Kupiec
Jan Pedersen
Penelope Sibun

Introduce

- What is a part-of-speech tagger system?
  - A system that uses context to assign parts of speech to words.
- What is the requirements of POS system?
  - Robust
  - Efficient
  - Accurate
  - Tunable
  - Reusable

Introduce

- What is the advantage of the POS system proposed in this paper?
  - Few resource (only a lexicon and some unlabeled training text)
  - Facilitate higher level analysis
  - Running in linear complexity

Brief of Methodology

- Approaches used in building text taggers
  - Rule-based approach (TAGGIT)
  - Rule-based approach with finite-state machines
  - Statistical methods (Markov model)

Brief of Methodology

- Types used in training
  - Make use of a tagged training corpus
    - Manually tag a small amount of text
    - Train a partially accurate model
    - Correct the tag in the text
    - Retrain the model
  - Without using tagged training corpus
    - Forward-Backward algorithm (Hidden Markov Model)

Hidden Markov Modeling

- Definision
  - HMM = (S, X, A, B, Π)
    - State Sequence: S= {S1, S2,…S_T} (Si $\in$ W, 1≤ i ≤ T)
    - Observation Sequence: X={X1, X2,…X_T} (1≤ i ≤ T)
    - Transfer Probability: A = {aij} (1≤ i, j ≤ N)

    where aij is the probability of moving from state i to state j
    - Output Probability: B = {bjk} (1≤ j ≤ N, 1≤ k ≤ M, M=|W|))

    where bjk is the probability of generating Sk at state j
    - Π = {Π i} where Π i is the probability of starting in state i

Hidden Markov Modeling

- Assumption for HMM
  - Assumption 1: Markov assumption-- probability of the occurrence of word S_i at time t depends only on occurrence of word S_{i-1} at time t-1. P(S1, S2,…Sn) can be estimated as $\prod_{i=1}^{n} P(Si \mid Si-1)$
  - Assumption 2: Output independent assumption--for all i, the S_i, X_i (observation sequence) are independent of all S_j and X_j, if i ≠ j. P(X1,…Xn|S1,…Sn) can be estimated as $\prod_{i=1}^{n} P(Xi \mid Si)$

Hidden Markov Modeling

- Before going on, let's see a small example
  - Let's say there are three types of weather: sunny , rainy , and foggy .
  - Precondition: A guy is locked in a room, the only piece of evidence is whether the person who comes in carries an umbrella  or not.
  - Task: ask about the weather outside based on the observations.

Hidden Markov Modeling

- Example continue
  - Finding the probability of a certain weather S∈ {sunny, Rainy, foggy} can only be based on the observation Xi (whether umbrella is carried). This conditional probability P(Si|Xi) can be rewritten according to Bayes's rule:
  - P(Si|Xi) = P(Xi|Si)P(Si)/P(Xi)
  - Because Xi is a known observation sequence, P(Xi) = 1, then P(Si|Xi) = P(Xi|Si)P(Si)= $\prod_{i=1}^{n} P(Si \mid Si-1) \prod_{i=1}^{n} P(Xi \mid Si)$

Hidden Markov Modeling

- Example continue
  - Task: Calculate the likelihood for the weather on these three days to have been {S1=sunny, S2=foggy, S3=sunny} and the person comes in without an umbrella (F).
  - P(S1=sunny,S2=foggy,S3=sunny|X1=F, X2=F, X3=F)

Today's weather	Tomorrow's weather		
	0.8	0.05	0.15
	0.2	0.6	0.2
	0.2	0.3	0.5

Weather	Probability of umbrella
Sunny	0.1
Rainy	0.8
Foggy	0.3

Hidden Markov Modeling

- Example continue

P(S1=sunny,S2=foggy,S3=sunny|X1=F, X2=F, X3=F)

= P(X1=F|S1=sunny) * P(X2=F|S2=foggy) * P(X3=F|S3=sunny) *P(S3=sunny|S2=foggy)* P(S2=foggy|S1=sunny) * P(S1=sunny)

= 0.9 * 0.7 *0.9 * 1/3 * 0.15 * 0.2

= 0.0057

So, The probability of {sunny, foggy, sunny} is 0.0057

Hidden Markov Modeling

- HMM in speech recognition
  - Task
    - Estimate the model parameters A, B and Π from a training set
    - Find the most likely sequence of words W given some acoustic input.

Hidden Markov Modeling

- HMM in speech recognition
  - Probability of the entire sequence
    - Forward probabilities: $\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(S_{t+1}) \quad 1 \le t \le T-1,$
    - Backward probabilities: $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(S_{t+1}) \beta_{t+1}(j) \quad 1 \le t \le T-1,$!
    - $P = \Sigma_i \alpha_t(i)\ \beta_t(i) = \Sigma_j \alpha_{t+1}(j)\ \beta_{t+1}(j)$
      $= \Sigma_i \alpha_t(i)\ (\Sigma_j a_{ij} b_j(S_{t+1})\ \beta_{t+1}(i)) = \Sigma_j (\Sigma_i \alpha_t(i)\ a_{ij} b_j(S_{t+1}))\beta_{t+1}(j)$
      $= \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(S_{t+1}) \beta_{t+1}(j)$

Hidden Markov Modeling

- HMM in speech recognition

Hence we can estimate a_{ij} by:

$$\hat{a}_{ij} = \frac{\gamma_{ij}}{\sum_{j=1}^{N} \gamma_{ij}} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(S_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i)}. \quad (3)$$

Similarly, b_{jk} and π_i can be estimated as follows:

$$\bar{b}_{jk} = \frac{\sum_{t \ni S_t = s_k} \alpha_t(j) \beta_t(j)}{\sum_{t=1}^{T} \alpha_t(j) \beta_t(j)} \quad (4)$$

and

$$\hat{\pi}_i = \frac{1}{P} \alpha_1(i) \beta_1(i). \quad (5)$$

Hidden Markov Modeling

- Viterbi Algorithm

To find the most probable such sequence we start by defining $\phi_1(i) = \pi_i b_i(S_1)$ for $1 \le i \le N$ and then perform the recursion

$$\phi_t(j) = \max_{1 \le i \le N} [\phi_{t-1}(i) a_{ij}] b_j(S_t) \quad (6)$$

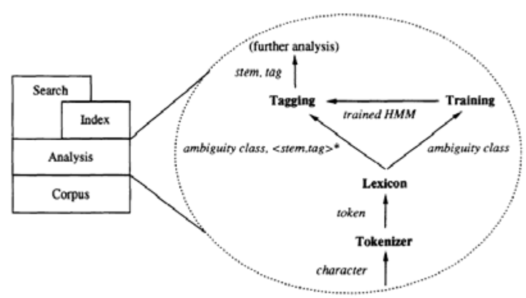
and

$$\psi_t(j) = \max_{1 \le i \le N}{}^{-1} \phi_{t-1}(i)$$

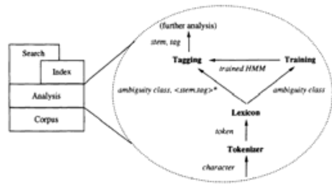
Notice: According to other reference book, $\Psi_t(j) = \max^{-1}[\phi_{t-1}(i)\ a_{ij}]$ rather than $\Psi_t(j) = \max^{-1} \phi_{t-1}(i)$

Architecture

- Tagger Modules in System Context



Architecture



- Five components
  - Term: a word stem annotated with part-of-speech
  - Corpus: provides text in a generic manner
  - Analysis: extracts terms from the text
  - Index: stores term occurrence statistics
  - Search: utilizes statistics to resolve queries

Architecture

- Tokenizer vs. Lexicon Implementation
  - Two basic classes of Tokenizer
    - Sentence boundary
    - Word
    - Other classes: numbers, paragraph boundaries ….
  - Three Stage of Lexicon
    - Construct lexicon manually
    - Guess ambiguity classes automatically (suffix)
    - Use default ambiguity classes

Evaluation

- Time Complexity: O(kTN)
- Space Complexity: T+2N(T+N+M+1)
- Accuracy: 96%
- Robust: fragments and ungrammaticalities
- Tunable: support tuning itself

  (various tagset, lexicon / empirical and prior information)

Application

- Phrase Recognition
- Word Sense Disambiguation
- Grammatical Function Assignment

Reference

- Barbara Resch, Hidden Markov Models-- A Tutorial for the Courses Computational Intelligence, http://www.igi.tugraz.at/lehre/CI