Chapter 8 Part-of-Speech Tagging

eight **parts of speech**: noun, verb, pronoun, preposition, adverb, conjunction, participle, and article.

Parts of speech (also known as **POS**, **word classes**, or **syntactic categories**) are useful because they reveal a lot about a word and its neighbors.Parts of speech are useful features for labeling **named entities** like people or organizations in **information extraction**,or for coreference resolution.

English Word Classes

Parts of speech can be divided into two broad supercategories: **closed class** types **open class** and **open class** types

- **closed classes**: thoes with relatively fixed membership.Closed class words are generally **function words** like *of*, *it*, *and*, or *you*, which tend to be very short, occur frequently, and often have structuring uses in grammar.
- **open classes**: new word are continually being created or borrowed

Four major open classes occur in the languages of the world: **nouns**, **verbs**, **adjectives**, and **adverbs**

The Penn Treebank Part-of-Speech Tagset

Part-of-Speech Tagging

- **part-of-speech tagging**: the process of assigning a part-of-speech marker to each word in an input text

- **Tagging**:**disambiguation** task; words are **ambiguous** —have more than one possible part-of-speech—and the goal is to fifind the correct tag for the situation.

Most Frequent Class Baseline: Always compare a classififier against a baseline at least as good as the most frequent class baseline (assigning each token to the class it occurred in most often in the training set)

HMM Part-of-Speech Tagging

The HMM is a **sequence model**. A sequence model or **sequence classi-fifier** is a model whose job is to assign a label or class to each unit in a sequence, thus mapping a sequence of observations to a sequence of labels.

- **Markov chain**: a model that tells us something about the probabilities of sequences of random variables, *states*, each of which can take on values from some set.

- **Markov assumption** on the probabilities of this sequence: that when predicting the future, the past doesn't matter, only the present.

  $Markov\ Assumption:\ P(q_i = a|q_1 \ldots q_{i-1}) = P(q_i = a|q_{i-1})$

The Hidden Markov Model

A **hidden Markov model** (**HMM**) allows us to talk about both *observed* events (like words that we see in the input) and *hidden* events (like part-of-speech tags) that we think of as causal factors in our probabilistic model.

The components of an HMM tagger

We compute the maximum likelihood estimate of this transition probability by counting, out of the times we see the fifirst tag in a labeled corpus, how often the fifirst tag is followed by the second: $P(t_i|t_{i-1}) = \frac{C(t_{i-1},t_i)}{C(_{i-1})}$

HMM tagging as decoding

- **decoding**： the task of determining the hidden variables sequence corresponding to the sequence of observations is called **decoding**.

The Viterbi Algorithm

As an instance of **dynamic programming**, Viterbi resembles the dynamic programming **minimum edit distance** algorithm of Chapter 2.

function VITERBI(*observations* of len *T*,*state-graph* of len *N*) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
for each state *s* **from** 1 **to** *N* **do** ; initialization step
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    $backpointer[s,1] \leftarrow 0$
for each time step *t* **from** 2 **to** *T* **do** ; recursion step
  for each state *s* **from** 1 **to** *N* **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

Working through an example

Extending the HMM Algorithm to Trigrams

Extending the algorithm from bigram to trigram taggers gives a small (perhaps a half point) increase in performance, but conditioning on two previous tags instead of one requires a signifificant change to the Viterbi algorithm.

The

maximum likelihood estimation of each of these probabilities can be computed from a corpus with the following counts:

$$Trigrams\ P(t_i|t_{i-1},t_{i-2}) = \frac{C(t_{i-2},t_{i-1},t_i)}{C(t_{i-2},t_{i-1})}$$
$$Bigrams\ P(t_i|t_{i-1}) = \frac{C(t_{i-1},t_i)}{C(t_{i-1})}$$
$$Unigrams\ P(t_i) = \frac{C(t_i)}{N}$$

- **deleted interpolation**： we successively delete each trigram from the training corpus and choose the λs so as to maximize the likelihood of the rest of the corpus.

Beam Search

function DELETED-INTERPOLATION(*corpus*) **returns** $\lambda_1, \lambda_2, \lambda_3$

  $\lambda_1, \lambda_2, \lambda_3 \leftarrow 0$
  foreach trigram t_1,t_2,t_3 with $C(t_1,t_2,t_3) > 0$
    depending on the maximum of the following three values
      case $\frac{C(t_1,t_2,t_3)-1}{C(t_1,t_2)-1}$: increment λ_3 by $C(t_1,t_2,t_3)$
      case $\frac{C(t_2,t_3)-1}{C(t_2)-1}$: increment λ_2 by $C(t_1,t_2,t_3)$
      case $\frac{C(t_3)-1}{N-1}$: increment λ_1 by $C(t_1,t_2,t_3)$
    end
  end
  normalize $\lambda_1, \lambda_2, \lambda_3$
  return $\lambda_1, \lambda_2, \lambda_3$

Unknown Words

To achieve high accuracy with part-of-speech taggers, it is also important to have a good model for dealing with **unknown words**.

We are thus computing for each suffix of length *i* the probability of the tag t_i given the suffix letters $P(t_i|l_{n-i+1} \ldots l_n)$. Back-off is used to smooth these probabilities with successively shorter suffixes.

Maximum Entropy Markov Models

we could turn logistic regression into a discriminative sequence model simply by running it on successive words, using the class assigned to the prior word, as a feature in the classification of the next word. When we apply logistic regression in this way, it's called the **maximum entropy Markov model** or **MEMM**.

In an MEMM, by contrast, we compute the posterior $P(T|W)$ directly, training it to discriminate among the possible tag sequences:

$$
\begin{aligned}
T &= argmax_T \ P(T|W) \\
&= argmax_T \prod_i P(t_i|w_i, t_{i-1})
\end{aligned}
$$

Features in a MEMM

A basic MEMM part-of-speech tagger conditions on the observation word itself, neighboring words, and previous tags, and various combinations, using feature **templates**

- **Word shape**: used to represent the abstract letter pattern of the word by mapping lower-case letters to 'x', upper-case to 'X', numbers to 'd', and retaining punctuation.

Decoding and Training MEMMs

The simplest way to turn logistic regression into a sequence model is to build a local classifier that classififies each word left to right, making a hard classifification on the fifirst word in the sentence, then a hard decision on the second word, and so on.

function GREEDY SEQUENCE DECODING(words W, model P) **returns** tag sequence T

for $i = 1$ **to** *length(W)*

$$\hat{t}_i = \underset{t' \in T}{\text{argmax}} \ P(t' \mid w_{i-l}^{i+l}, t_{i-k}^{i-1})$$

Instead we decode an MEMM with the **Viterbi** algorithm just as with the HMM, fifinding the sequence of part-of-speech tags that is optimal for the whole sentence.

Bidirectionality

These are names for situations when one source of information is ignored because it is **explained away** by another source.One way to implement bidirectionality is to switch to a more powerful model called a **conditional random fifield** or **CRF**.Simpler methods can also be used; the **Stanford tagger** uses a bidirectional version of the MEMM called a cyclic dependency network

Part-of-Speech Tagging for Morphological Rich Languages