Part of speech tagging

Part of speech

A category of words that play similar roles within the syntactic structure of a sentence.

Part of speech tagging

Part of speech tagger = program that tags each word in sentence with its part of speech.

Can be approached using supervised learning (requires training data).

Ambiguity (words can have different tags) → combinatorial explosion

Accuracy

Diagonal / whole

Precision

Exact match / column

Recall

Exact match / row

Hidden Markov model (HMM)

Words have probabilities tied to each of its tags (jag → NN, jag → PN)

Tags have probabilities for its next tag (NN → VB)

HMM has two probabilities: transitional (tag2 given tag1) and output (word given tag).

Transitional first, then output at every junction.

P(VB | PN) → amount of PN followed by VB / all occurences of VB

P(jag | PN) → amount of jag when PN / all words that are PN

Multi-class perceptron

Feature window

HMM looks back once; might want to look further, or look forward. But dont want to see too much (efficiency).

Need a feature window. Feature window sees x in front and x in back of the current word.

Evaluate a part-of-speech tagger based on accuracy, precision, and recall

Compute the probability of a tagged sentence in a hidden Markov model

Probability of tagged sentence → product of transition and output (transition * output)

Syntactic Analysis

Phrase structure tree

Sentence divides into parts (S → Noun Phrase, Verb Phrase), which in turn divide into parts (NP → Pro → “I”, VP → Verb, NP).

Dependency tree

“This word depends on that word”. Verbs have subjects and objects, etc.

Probabilistic context-free grammar (PCFG)

Words within sentences form phrases:
“Kim read [a book]”, “Kim read [a very interesting book about grammar]”

Syntactic head → most important word in sentence.

Context free grammar → Phrases combine. How to combine? Context free grammar!
Example: Sentence → NP, VB (Basically BNF)

Probabilistic → Number of trees grows exponentially with length of sentence. Not all parse trees are relevant, only most probable.

PCFG → Every rule R has probability P(R), and sum of all P(R) with same left side is 1.

Tree probability = product of all P(R)

Transition-based dependency parser

Contains: buffer, stack, tree

Operations:

- Shift transition → Pop buffer, push to stack
- Left arc transition → Dependency from top of stack to second top, remove second top.
- Right arc transition → Dependency from second top of stack to top, remove top.

Terminate when buffer is empty and stack has 1 or less elements

Learn a probabilistic context-free grammar from a treebank

Estimate rule probabilities → count of specific rule / count of all rules with same left side.

Simulate a transition-based dependency parser

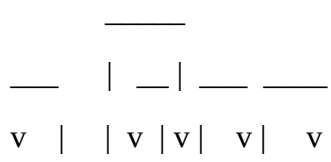
Stack → ← buffer

[] [I booked a flight from L.A.]

I booked a flight from L.A.

EVENTUALLY

[booked] []



I booked a flight from L.A.