Part-of-Speech Tagging: Fundamentals, Techniques, and Applications

Natural Language Processing Course

Outline

What is Part-of-Speech Tagging?

- **Definition**: the process of assigning a grammatical category (or part of speech) to each word in a sentence based on how it is used in context.
- **Goal**: IPOS tagging helps computers understand the role and function of each word in a sentence, such as whether a word is the subject, action, or description. For example, the word "run" can be a verb ("I run every morning") or a noun ("It was a good run"). POS tagging ensures that the correct tag is assigned based on the sentence structure and meaning.
- **Example**:
  - Text: "The cat sits on the mat"
  - Tagged: The/DET cat/NOUN sits/VERB on/PREP the/DET mat/NOUN

Why POS Tagging Matters

By helping computers recognize the role of each word, POS tagging improves the accuracy of text-based applications such as:

- **Applications**:
  - Text-to-Speech systems
  - Machine Translation
  - Information Extraction
  - Question Answering
  - Grammar Checking
- **Foundation** for:
  - Syntactic parsing
  - Named Entity Recognition
  - Semantic analysis

Penn Treebank Tag Set Overview

- linguistically annotated corpus
- Widely used standard in NLP
- Contains 36 POS tags and 12 other tags
- More granular than basic POS categories

Wall Street Journal (WSJ) articles (financial news), Brown Corpus (various genres of American English), Transcriptions of spoken conversations and telephone speech

Noun Tags in Penn Treebank

Basic Noun Tags:

- NN: Singular noun (book, tree)
- NNS: Plural noun (books, trees)
- NNP: Proper noun (John, London)
- NNPS: Plural proper noun (Vikings)

Example:

- "The book/NN is on the shelf/NN"
- "The books/NNS are many"
- "John/NNP lives in Paris/NNP"

Verb Tags in Penn Treebank

- **VB**: Base form (take)
- **VBD**: Past tense (took)
- **VBG**: Gerund/present participle (taking)
- **VBN**: Past participle (taken)
- **VBP**: Non-3rd singular present (take)
- **VBZ**: 3rd singular present (takes)
- **Example**: "He/PRP takes/VBZ and has/VBZ taken/VBN"

Other Important Penn Treebank Tags

Adjectives & Adverbs:

- JJ: Adjective
- JJR: Comparative
- JJS: Superlative
- RB: Adverb
- RBR: Comparative
- RBS: Superlative

Function Words:

- IN: Preposition
- DT: Determiner
- CC: Coordinating conjunction
- TO: 'to'
- PRP: Personal pronoun
- PRP$: Possessive pronoun

How POS tagging works

POS tagging uses rules or algorithms to analyze the words and their surrounding context.
It determines:

- What part of speech a word belongs to (like a noun or verb).
- How it relates to other words in the sentence.

Rule-Based Techniques

Components of Rule-Based Systems:

- **Lexicon**:
  - Word-tag dictionary
  - Morphological rules
  - Exception lists
- **Disambiguation Rules**:
  - Context rules (previous and following rules)
  - Patterns rule (ET + [target word] + NOUN $\rightarrow$ target word is likely adjective)
  - Priority ordering (most specific rules first, then, contextual rules, morphological rules)
- **Example Rule**:
  - "If word ends in 'ing' and previous word is 'be/am/is/are' $\rightarrow$ VBG"

Rule-based technique: example

Example: *She is reading a book*

- `she`: PRP
- `is`: VBZ
- `reading`: VB/VBG
- `a`: DT
- `book`: NN/VB

"is" is a helping verb, so "reading" is tagged as VBG (present participle). "a" is a determiner, so "book" is tagged as NN (noun).

Final POS:

- `she`: PRP
- `is`: VBZ
- `reading`: VBG
- `a`: DT
- `book`: NN

Probabilistic method: Hidden Markov Model

Key Components:

- **States**: POS tags
- **Observations**: Words
- **Probabilities**:
  - Transition: $P(tag_i|tag_{i-1})$ (Transition probabilities: how likely is it to move from one state to another)
  - Emission: $P(word_i|tag_i)$ (Emission probabilities: how likely is it to observe something in each state)

Probabilistic method: Hidden Markov Model

Training Process:

- Use probabilistic models trained on labeled data (penn treebank)
- Count tag transitions and calculate emission probabilities

Viterbi Decoding:

- Dynamic programming algorithm
- During the tagging process, the system finds the most probable sequence of tags for a given sentence using Viterbi algorithm
- Time complexity: $O(NT^2)$
  - N = sentence length
  - T = number of possible tags

Hidden Markov Model: example

Example: *She eats an apple.*

Possible Tags for Words:

- `she`: PRP
- `eats`: VBZ / NNS
- `an`: DT
- `apple`: NN

Transition and Emission Probabilities:

- $P(\text{PRP} \mid \text{START}) = 0.6$
- $P(\text{VBZ} \mid \text{PRP}) = 0.6$
- $P(\text{NNS} \mid \text{PRP}) = 0.1$
- $P(\text{DT} \mid \text{VBZ}) = 0.5$
- $P(\text{NN} \mid \text{DT}) = 0.7$

Using Viterbi algorithm that finds the most probable sequence of tags:

`PRP VBZ DT NN`

Neural Approaches

Bidirectional LSTM:

- Word embeddings input
- Forward and backward states
- Character-level features
- CRF output layer

Example: *She eats an apple*
Input: Word embeddings of [“she”, “eats”, “an”, “apple”]
BiLSTM Output: A context-aware vector for each word.
Output: [“DT”, “NN”, “VBD”, “DT”, “NN”]

Evaluating POS Taggers

- **Metrics**:
  - Accuracy
  - Per-tag precision/recall
  - Confusion matrix
- **Common Benchmarks**:
  - Penn Treebank
  - Universal Dependencies

Model Performance

Accuracy Comparison:

- Rule-based: 90-95%
- HMM: 95-96%
- BiLSTM-CRF: 97-98%

NLTK Example

```
import nltk
from nltk import pos_tag
from nltk.tokenize import word_tokenize

text = "The quick brown fox jumps over the lazy dog"
tokens = word_tokenize(text)
tagged = pos_tag(tokens)
print(tagged)
# Output: [('The', 'DT'), ('quick', 'JJ'),
# ('brown', 'JJ'), ('fox', 'NN'),...]
```

Common Challenges

- **Ambiguity**:
  - "Bank" as noun or verb
  - "Desert" as noun or verb
- **Unknown Words**:
  - Proper nouns
  - Technical terms
  - Neologisms
- **Context Dependency**:
  - "Lead" (verb/noun)
  - "Close" (verb/adjective)

Exercise: Basic POS Tagging

How does tokenization affect the accuracy of POS tagging? Handling Contractions and Handling Compound Words.

Do you think POS tagging models trained on general corpora like Penn Treebank can perform well on specialized domains (e.g., legal, medical)? Why? Many domain-specific terms are absent in general corpora.

Questions?

Thank you for your attention!

Time for questions and discussion