Name: Sanket S. Fulzele

Roll No: 371018

PRN: 22110728

Div: A

POS Tagging In Different Languanges

Part-of-Speech (PoS) tagging involves labeling words in a sentence with their grammatical categories, such as nouns, verbs, adjectives, and so on. Different languages have their own PoS tag sets tailored to their grammatical structures and linguistic nuances. Here's a brief overview of PoS tag sets for some languages:

1. **English**:
    - Universal PoS Tag Set (Universal Dependencies): This tag set includes tags like NOUN (noun), VERB (verb), ADJ (adjective), ADV (adverb), PRON (pronoun), DET (determiner), ADP (adposition), CONJ (conjunction), and more. It aims to provide a universal standard for PoS tagging across languages.
2. **Spanish**:
    - Universal Dependencies for Spanish: Similar to English, Spanish also follows the Universal PoS Tag Set with tags like NOUN, VERB, ADJ, ADV, PRON, DET, ADP, CONJ, etc. Additionally, Spanish has specific tags for reflexive verbs (REFL), gerunds (GRND), and other grammatical constructs.
3. **French**:
    - Universal Dependencies for French: French PoS tags include similar categories to English and Spanish but may have additional tags for gender and number agreement (e.g., NOUN.FEM for feminine nouns, NOUN.PL for plural nouns).
4. **German**:
    - Universal Dependencies for German: German PoS tags also align with the universal tag set but may include specific tags for case markings (e.g., NOUN.NOM for nominative case, NOUN.ACC for accusative case).

Types of POS Tagging

Part-of-Speech (PoS) tagging is a fundamental task in natural language processing (NLP) that involves labeling words in a text with their corresponding grammatical categories, such as nouns, verbs, adjectives, adverbs, etc. Several approaches are used for PoS tagging, each with its strengths and limitations. Here's a brief discussion of some common approaches:

1. **Rule-Based Tagging**:
    - **Dictionary Lookup**: This approach involves using a dictionary of words along with their associated PoS tags. Words are looked up in the dictionary to assign tags based on predefined rules.
    - **Rule-Based Patterns**: Rules and patterns are defined based on linguistic knowledge and regularities in language. For example, words ending in "-ing" are often verbs, and words starting with capital letters are likely proper nouns.
2. **Statistical Tagging**:
    - **N-gram Models**: These models use sequences of N words (e.g., bigrams, trigrams) to predict the PoS tag of a word based on the context provided by surrounding words.
    - **Hidden Markov Models (HMMs)**: HMMs model the probability of transitioning between PoS tags and the probability of observing words given their tags, learning from annotated training data.
    - **Maximum Entropy Models**: These models assign probabilities to PoS tags based on features extracted from the context, such as neighboring words, prefixes, suffixes, and word shapes.
3. **Machine Learning-Based Tagging**:
    - **Supervised Learning**: Utilizes annotated training data to train classifiers (e.g., decision trees, support vector machines, neural networks) to predict PoS tags based on features derived from the text.
    - **Semi-Supervised Learning**: Combines labeled and unlabeled data to improve tagging accuracy, often using techniques like self-training or co-training.

- **Deep Learning Models**: Neural network architectures such as recurrent neural networks (RNNs), long short-term memory (LSTM) networks, and transformer models (e.g., BERT) have shown state-of-the-art performance in PoS tagging by learning complex contextual representations.

4. **Hybrid Approaches**:
   - **Combining Rules and Statistics**: Hybrid models integrate rule-based mechanisms with statistical or machine learning components to benefit from both linguistic knowledge and data-driven learning.
   - **Ensemble Methods**: Combining predictions from multiple taggers, each using a different approach or model, can improve overall tagging accuracy and robustness.

Each approach has its advantages and is suited to different scenarios based on available data, computational resources, and the desired balance between linguistic accuracy and scalability. Modern NLP systems often employ a combination of these approaches to achieve high-performance PoS tagging.

1. Inbuilt PoS Tagging Using Libraries

```
import nltk
nltk.download("punkt")
nltk.download('averaged_perceptron_tagger')
from textblob import TextBlob

# Using NLTK
text = "The quick brown fox jumps over the lazy dog"
tokens = nltk.word_tokenize(text)
pos_tags_nltk = nltk.pos_tag(tokens)
print("NLTK PoS Tags:", pos_tags_nltk)

# Using TextBlob
blob = TextBlob(text)
pos_tags_textblob = blob.tags
print("TextBlob PoS Tags:", pos_tags_textblob)
```

```
NLTK PoS Tags: [('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox', 'NN'), ('jumps', 'VBZ'), ('over', 'IN'), ('the', 'DT'), ('l
TextBlob PoS Tags: [('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox', 'NN'), ('jumps', 'VBZ'), ('over', 'IN'), ('the', 'DT'),
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

2. Regular Expressions for PoS Categories

```
from nltk import RegexpTagger

patterns = [
    (r'.*ing$', 'VBG'),
    (r'.*ed$', 'VBD'),
    (r'.*es$', 'VBZ'),
    (r'.*ould$', 'MD'),
    (r'.*\'s$', 'NN$'),
    (r'.*s$', 'NNS'),
    (r'^-?[0-9]+(\.[0-9]+)?$', 'CD'),
    (r'the', 'DT'),
    (r'in', 'IN'),
    (r'.*', 'NN')  # Default: nouns
]

regexp_tagger = RegexpTagger(patterns)

# Example sentence
sentence = "5 friends have been singing in the rain"
tagged_words = regexp_tagger.tag(sentence.split())
print(tagged_words)
```

```
[('5', 'CD'), ('friends', 'NNS'), ('have', 'NN'), ('been', 'NN'), ('singing', 'VBG'), ('in', 'IN'), ('the', 'DT'), ('rain', 'NN')]
```

3. Dictionary/Lookup Table for PoS Tagging

```
pos_dict = {
    'dog': 'NN',        # Noun
    'jump': 'VB',       # Verb
    'quick': 'JJ',      # Adjective
    'lazy': 'JJ',       # Adjective
    'brown': 'JJ',      # Adjective
    'fox': 'NN',        # Noun
    'over': 'IN',       # Preposition
    'the': 'DT',        # Determiner
    'tree': 'NN',       # Noun
    'eats': 'VB',       # Verb
    'apple': 'NN',      # Noun
    'and': 'CC',        # Conjunction
    'orange': 'NN',     # Noun
    'juice': 'NN',      # Noun
    'big': 'JJ',        # Adjective
    'small': 'JJ',      # Adjective
    'quickly': 'RB',    # Adverb
    'quietly': 'RB',    # Adverb
    'very': 'RB',       # Adverb
    'eat': 'VB'         # Verb
}

def pos_tag_with_dict(sentence):
    tokens = nltk.word_tokenize(sentence)
    pos_tags = [(token, pos_dict.get(token.lower(), 'NN')) for token in tokens]
    return pos_tags

# Example usage:
text_to_tag = "The quick brown fox jumps over the lazy dog"
pos_tags_dict = pos_tag_with_dict(text_to_tag)
print("PoS Tags with Dictionary:", pos_tags_dict)
```

```
PoS Tags with Dictionary: [('The', 'DT'), ('quick', 'JJ'), ('brown', 'JJ'), ('fox', 'NN'), ('jumps', 'NN'), ('over', 'IN'), ('the',
```

4. Train Unigram, Bi-gram, and Trigram Taggers

```
from nltk.corpus import brown
nltk.download('brown')
from nltk.tag import UnigramTagger, BigramTagger, TrigramTagger

# Training Data
tagged_sents = brown.tagged_sents(categories='news')

# Unigram Tagger
unigram_tagger = UnigramTagger(tagged_sents)

# Bigram Tagger
bigram_tagger = BigramTagger(tagged_sents)

# Trigram Tagger
trigram_tagger = TrigramTagger(tagged_sents)

# Testing
test_sentence = "The cat sat on the mat."
tokens = nltk.word_tokenize(test_sentence)
print("Unigram Tagger:", unigram_tagger.tag(tokens))
print("Bigram Tagger:", bigram_tagger.tag(tokens))
print("Trigram Tagger:", trigram_tagger.tag(tokens))
```

```
[nltk_data] Downloading package brown to /root/nltk_data...
[nltk_data]   Package brown is already up-to-date!
Unigram Tagger: [('The', 'AT'), ('cat', None), ('sat', 'VBD'), ('on', 'IN'), ('the', 'AT'), ('mat', 'NN'), ('.', '.')]
Bigram Tagger: [('The', 'AT'), ('cat', None), ('sat', None), ('on', None), ('the', None), ('mat', None), ('.', None)]
Trigram Tagger: [('The', 'AT'), ('cat', None), ('sat', None), ('on', None), ('the', None), ('mat', None), ('.', None)]
```