Parts of Speech Tagging with Natural language Processing (NLP)

ONUBA CHIBUIKE WINNER*

* onubawinner042@gmail.com

INTRODUCTION

Part of speech tagging, also known as POS tagging or word class tagging, is the process of assigning a grammatical category to each word in a text, such as noun, verb, adjective, adverb, etc.

For example,
In the sentence "She likes to read books", the POS tags are:
She - pronoun
likes - verb
to - preposition
read - verb
books - noun

POS tagging is useful for many downstream NLP applications, such as syntactic parsing, semantic analysis, information extraction, machine translation, sentiment analysis, and more. By knowing the POS tags of words, we can better understand the structure and meaning of sentences, and perform more accurate and efficient processing.

APPLICATIONS

With POS tagging, NLP algorithms can:

- **Enhance Text Understanding**: By identifying the roles of words in a sentence, machines can grasp context more accurately, leading to better comprehension of human language.
- **Improve Search & Information Retrieval**: POS tagging enables more precise indexing and retrieval of information, making search engines smarter and more efficient.
- **Facilitate Sentiment Analysis**: Understanding the grammatical structure helps in extracting sentiment from text, empowering businesses to gauge customer opinions more effectively.
- **Enable Machine Translation**: Accurate POS tagging is crucial for translating text between languages, ensuring grammatical correctness and preserving meaning.
- **Support Text Summarization & Generation**: POS tagging aids in identifying key information in a text, facilitating summarization and generation of concise, coherent content.

WORKFLOW

QUICK FOX JUMPS

TOKENIZER — Divide the input text into discrete tokens

- **“QUICK”**
- **“FOX”**
- **“JUMPS”**

Language Models (e.g., NLTK or SpaCy)

These models provide a foundation for understanding a language’s grammatical structure

TAGGING

- **“QUICK”** → Adjective (JJ)
- **“FOX”** → Noun (NN)
- **“JUMPS”** → Verb (VBZ)

There are different methods and tools for POS tagging, ranging from rule-based systems to statistical models to neural networks. Some of the popular POS taggers are NLTK, spaCy, Stanford CoreNLP, and Transformers. Each of these tools has its own advantages and disadvantages, such as speed, accuracy, coverage, and language support.

NLTK

NLTK is a widely used library for NLP, which provides a simple interface for POS tagging. To use NLTK, we need to import the library, tokenize the sentence, and apply the `pos_tag` function. The output is a list of tuples, where each tuple contains a word and its corresponding POS tag.

POS Tagging with NLTK

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


# Example sentence
sentence = "The quick brown fox jumps over the lazy dog."

# Tokenize the sentence
tokens = word_tokenize(sentence)

# Perform POS tagging
pos_tags = pos_tag(tokens)

# Print the tagged words
for word, tag in pos_tags:
    print(f"{word} | {tag}")
```

[7] ✓ 0.0s

```
The | DT
quick | JJ
brown | NN
fox | NN
jumps | VBZ
over | IN
the | DT
lazy | JJ
dog | NN
. | .
```

SPACY

spaCy is another popular library for NLP, which offers a fast and accurate POS tagger. To use spaCy, we need to import the spacy library, load a language model, and create a Doc object from the sentence. The output is an iterable of Token objects, which have attributes such as text and pos_.

POS Tagging with SpaCy

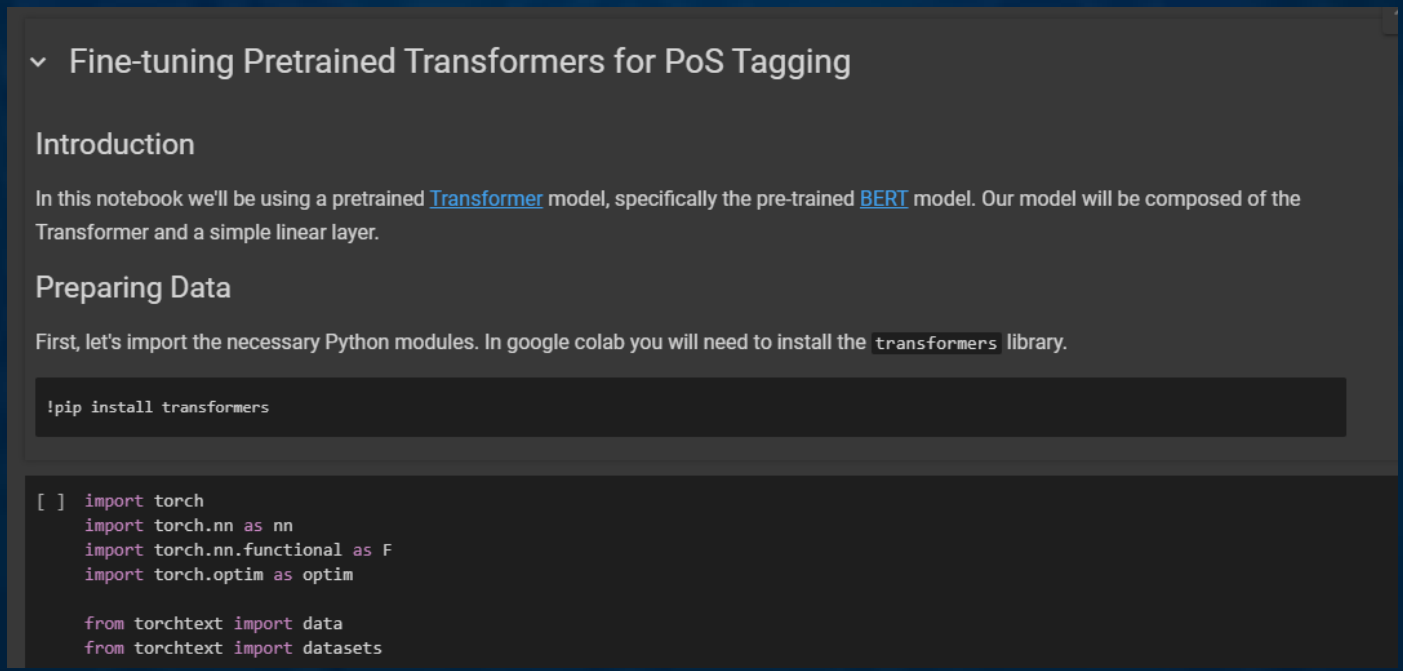
```
import spacy

# load the english language model
nlp = spacy.load("en_core_web_sm")

sentence = "She likes to read books"

# create a doc object from the sentence
doc = nlp(sentence)


for token in doc:
    print(token.text, '|', token.pos_, '|', spacy.explain(token.pos_))
```

[9] ✓ 6.9s

```
She | PRON | pronoun
likes | VERB | verb
to | PART | particle
read | VERB | verb
books | NOUN | noun
```

Transformers

Transformers is a library for NLP that provides easy access to pre-trained models for various NLP tasks, such as POS tagging. To use Transformers, we need to import the library, load a pipeline object, and pass the sentence as an argument. The output is a list of dictionaries, where each dictionary contains a word and its corresponding POS tag.



Fine-tuning Pretrained Transformers for PoS Tagging

Introduction

In this notebook we'll be using a pretrained Transformer model, specifically the pre-trained BERT model. Our model will be composed of the Transformer and a simple linear layer.

Preparing Data

First, let's import the necessary Python modules. In google colab you will need to install the `transformers` library.

```
!pip install transformers
```

```
[ ]  import torch
     import torch.nn as nn
     import torch.nn.functional as F
     import torch.optim as optim

     from torchtext import data
     from torchtext import datasets
```

View notebook **here**

Method	Advantage
NLTK	Simple and easy to use, widely adopted, supports many languages and tag sets
spaCy	Fast and accurate, integrates well with other NLP components, provides rich linguistic features
Stanford CoreNLP	Comprehensive and robust, covers a wide range of NLP tasks, offers high-quality annotations
Transformers	Powerful and versatile, provides access to pre-trained models, supports various NLP tasks