CY SHIELD

POS with NetworkX in articles

Overview

The goal of this project is to develop a part-of-speech (POS) tagger for Arabic text using a Conditional Random Field (CRF) algorithm. POS tagging is a fundamental task in natural language processing that involves assigning grammatical tags to words in a given text. By accurately predicting POS tags, we can gain insights into the syntactic structure of sentences and present it by NetworkX.

Data set

This dataset was created to support part of speech (POS) tagging in dialects of Arabic. It contains sets of 350 manually segmented and POS tagged tweets .

Data Instances

- `Fold`: 4

- `SubFold`: A

- `Word`: [قلبك ,من ,حد ,تحب ,لما ,ليه, ...]

- `Segmentation`: [قلب+ك ,من ,حد ,تحب ,لما ,ليه, ...]

- `POS`: [PART, PART, V, NOUN, PREP, NOUN+PRON, ...]

It consists of 100 instances with various Arabic sentences and their corresponding POS tags

1.Preprocessing

- The data was received from an API by sending an HTTP GET request to the specified URL. The API is expected to return data in JSON format. The response from the API is then processed to extract the relevant information.

- The loaded JSON data is typically structured with different fields or attributes. In this case, the `features` and `rows` fields are extracted from the `data` dictionary. These fields contain the relevant information needed for further processing.

- Finally, the extracted data is converted into a pandas DataFrame using the `pd.DataFrame()` constructor. This allows for easier manipulation, analysis, and visualization of the data. The `head()` method is used to display the first few rows of the DataFrame, while the `shape` attribute provides the number of rows and columns in the DataFrame.

- Data was already tokenized; it contains the individual words or tokens that make up the sentence. The words are represented as a list, such as [قلبك, من, حد, تحب, لما, ليه, ...]. Each word represents a distinct unit of meaning in the sentence.

- The POS tags are assigned to each word and are represented as a list, such as [PART, PART, V, NOUN, PREP, NOUN+PRON, ...]. Each POS tag provides information about the word's syntactic function, such as verb, noun, preposition, etc. The + symbol is used to denote a combination of POS tags for morphologically complex words.

2.Data Preparation

- **Step 1**: extract the **'words'** column from the DataFrame `df` and convert it into a Python **list**. Each element in the list represents a sentence or sequence of words. Similarly, extract the **'pos_tags'** column from the DataFrame `df` and convert it into a Python **list**. Each element in the list represents the corresponding part-of-speech tags for the words in the respective sentence.

- **Step3**: combine the `**words_list**` and `pos_tags_list` lists element-wise. The resulting pairs of word and part-of-speech tags are then converted into a new list called `data`. Each element in `data` is a tuple containing a word and its corresponding part-of-speech tag.

- **Step3**: **iterates** over each element (sentence) in the `**train_data**` list. For each sentence, the `**zip()**` function is used to **combine** the words and **part-of-speech** tags into tuples, creating a nested list of tuples. This is done for all sentences in `train_data`. The resulting `train_data_flat` is a flattened list of tuples, **where each tuple contains a word and its corresponding part-of-speech tag**.

These steps are part of the data preparation process for training a CRF (Conditional Random Field) tagger. By splitting the data into training and testing sets, and converting them into the appropriate format of word-tag tuples, the data is ready to be used for training and evaluating the CRF tagger model.

3.CRF (Conditional Random Field) tagger

- The CRF (Conditional Random Fields) model is a popular probabilistic graphical model used for sequence labeling tasks, such as part-of-speech tagging. It models the conditional probability of a sequence of labels given a sequence of input observations.

- In the provided code snippet, the `CRFTagger` class is used to train a CRF tagger model on the `train_data_flat` dataset.

- During training, the CRF tagger model learns the transition probabilities between different labels (part-of-speech tags) and the emission probabilities of observing certain words given specific labels. It estimates these probabilities based on the provided training data using the maximum likelihood estimation or other optimization algorithms.

- After the training process is completed, the `crf_tagger` object contains the trained CRF tagger model, which can be used to predict part-of-speech tags for new, unseen data.

5

Predicting

- unseen data is preprocessed and tokenized, and then the trained CRF Tagger model is used to predict part-of-speech (POS) tags for each token. Here's an explanation of the steps involved:

Text:= "لعب الولد الكرة في المدرسة الكبيرة"

```
لعب ADJ
الولد NOUN
الكرة NOUN+NSUFF
في PREP
المدرسة NOUN+NSUFF
الكبيرة ADJ+NSUFF
```

Text = "أرحب بيكم"

```
أرحب V
بيكم PREP+PRON
```

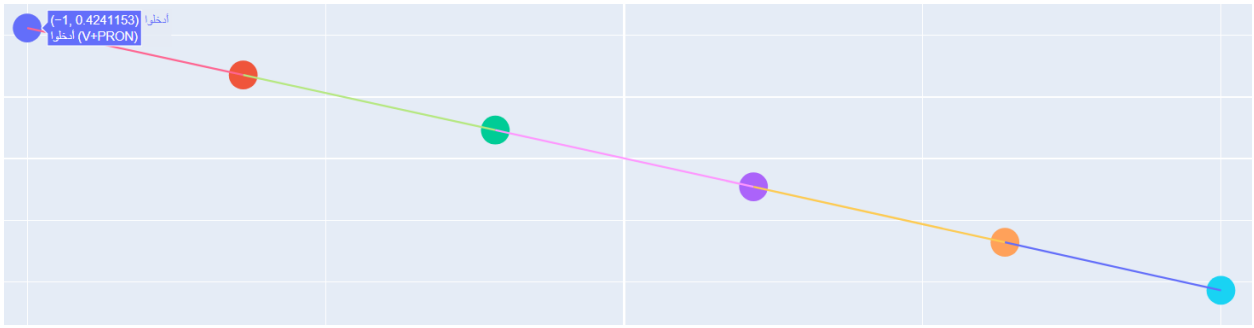
Text = " ادخلوا مصر ان شاء الله آمنين "

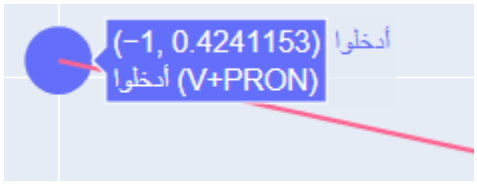
```
أدخلوا V+PRON
مصر NOUN
ان PART
شاء NOUN
الله NOUN+NSUFF
امنين ADJ+NSUFF
```

NetworkX and Plotly

visualizing a network of POS tags using NetworkX and Plotly.

The process begins by creating an empty graph using NetworkX, where each POS tag is represented as a node in the graph. Edges are then added between adjacent tags to capture the relationships. The layout of the graph is determined using a spring layout algorithm. Using Plotly, a figure is created to display the network visualization. Nodes are represented as markers with their positions determined by the layout, and edges are displayed as lines connecting the nodes. Finally, the figure is shown to visualize the network of POS tags.





Tools Used:

- Python: Programming language used for implementing the CRF tagger and performing data processing.
- NLTK: Python library used for natural language processing tasks, including tokenization and training the CRF tagger.
- NetworkX: Python library used for creating and manipulating the graph structure of the POS tags.
- Plotly: Python library used for visualizing the graph of the POS tags.

External Resources:

- Arabic POS Dialect dataset: Obtained from a Hugging Face API, consisting of Arabic sentences and their POS tags.

Challenges Faced:

The biggest challenge faced during this project was obtaining an Arabic POS dataset suitable for training the tagger. Limited availability of annotated Arabic text data posed difficulties in finding a comprehensive and high-quality dataset. However, the Arabic POS Dialect dataset provided a sufficient number of instances for training and evaluation.

Key Learnings:

Through this project, I learned about the process of developing a POS tagger using a CRF algorithm. I gained experience in data preprocessing,and evaluating NLP models.

Additionally, I improved my understanding of Arabic language processing and the challenges associated with it..

Overall, this project enhanced my knowledge of NLP techniques and their practical application to Arabic text analysis.