Name: Kenny Patel | **Roll No.** : 19074010

Part-of-Speech Tagging, Chunking and Named Entity Recognition using CRF++

CRF++ is a simple, customizable, and open source implementation of Conditional Random Fields (CRFs) for segmenting/labeling sequential data. CRF++ is designed for generic purpose and will be applied to a variety of NLP tasks, such as Named Entity Recognition, Information Extraction and Text Chunking.

Chunking

Chunking is a process of extracting phrases from unstructured text. Instead of just simple tokens which may not represent the actual meaning of the text, it's advisable to use phrases such as “South Africa” as a single word instead of ‘South’ and ‘Africa’ separate words. Chunking works on top of POS tagging, it uses pos-tags as input and provides chunks as output. Similar to POS tags, there are a standard set of Chunk tags like Noun Phrase(NP), Verb Phrase (VP), etc. Chunking is very important when you want to extract information from text such as Locations, Person Names etc. In NLP it is called Named Entity Extraction.

- Corpus : Training and testing data is downloaded from CoNLL shared tasks 2000.
- Template : Template is the key to use CRF++. It can help us automatically generate a series of feature functions, instead of generating our own feature functions, which is one of the core concepts of CRF algorithms. Each of the three folders of chunking consists of templates with different features based on which the model is trained.
- Training : A model is generated for each template by using the templates and training data with the following command:

```
crf_learn template_file train.data model_name
```

- Forecast : After training the model, we can use the trained model to predict the new data. The format of the prediction command is as follows:

```
crf_test -m model_name test.data > chunking_output.txt
```

- Accuracy : accuracy_chunking.py consists of the python script used to check the accuracy of the output by comparing the last two columns of the chunking output file.

Accuracy from the three Models is as follows:

1. accuracy: 96.06349072334677
2. accuracy: 96.12892331722144
3. accuracy: 95.55691580302678

POS Tagging

POS or part-of-speech tagging is the technique of assigning special labels to each token in text, to indicate its part of speech, and usually even other grammatical connotations, which can later be used in text analysis algorithms. For example, for the sentence -

She is reading a book.

‘She’ would use the POS tag of the pronoun, ‘is’ would get an article tag, ‘reading’ a verb tag, ‘a’ would get an article tag and ‘book’ would get a noun tag. We can then do a search for all verbs which would pull up the word reading, and also use these tags in other algorithms.

- Template : Template is the key to use CRF++. It can help us automatically generate a series of feature functions, instead of generating our own feature functions, which is one of the core concepts of CRF algorithms. Each of the three folders of POS_tagging consists of templates with different features based on which the model is trained.

- Training : A model is generated for each template by using the templates and training data with the following command:

```
crf_learn template_file train.data model_name
```

- Forecast : After training the model, we can use the trained model to predict the new data. The format of the prediction command is as follows:

```
crf_test -m model_name test.data > tagging_output.txt
```

- Accuracy : accuracy_tagging.py consists of the python script used to check the accuracy of the output by comparing the last two columns of the tagging output file.

Accuracy from the three Models is as follows:

1. accuracy: 93.04514848977352
2. accuracy: 93.42719040884818
3. accuracy: 93.20134242353885

Named Entity Recognition (NER)

Named Entity Recognition (NER) is an important basic tool in the fields of information extraction, question answering system, parsing, machine translation and so on. It plays an important role in the practical process of natural language processing technology.

- Template : Template is the key to use CRF++. It can help us automatically generate a series of feature functions, instead of generating our own feature functions, which is one of the core concepts of CRF algorithms. Each of the three folders of NER consists of templates with different features based on which the model is trained.

- Training : A model is generated for each template by using the templates and training data with the following command:

```
crf_learn template_file train.data model_name
```

- Forecast : After training the model, we can use the trained model to predict the new data. The format of the prediction command is as follows:

```
crf_test -m model_name test.data > ner_output.txt
```

- Accuracy : accuracy_ner.py consists of the python script used to check the accuracy of the output by comparing the last two columns of the tagging output file.

Accuracy from the three Models is as follows:

1. accuracy: 96.64585676063437
2. accuracy: 96.69238822753887
3. accuracy: 94.77684283997053