DEMO 3:

Part-of-Speech Tagging

Team Members:

Prashanth ASOK KUMAR

Annanya CHITRA KANNAN

TABLE OF CONTENTS

Pre-Requisites:

We will be making Part of Speech Tagging using the OpenNMT for the machine learning approach.

Install the OpenNMT and Subwords in the computer using pip command:

pip install OpenNMT-tf
pip install subword-nmt

Since we faced few issues while training the model, we made use of the below command to setup the environment as in the github link
https://github.com/QwantResearch/opennmt-tf

pip install git+https://github.com/QwantResearch/opennmt-tf

We will be making use of the steps and guidelines provided in the below link to perform PoS using OpenNMT.

https://opennmt.net/OpenNMT-tf/

The train and test data which we will be using to train the data is obtained from the link below.

https://cservan.github.io/data/tp3_files.tgz

Also, the sample configuration and model which will be used in this project is obtained from the link below:

https://cservan.github.io/tools/seq_tagger_data.tar.gz

Creating data:

The data present in the input data set is not in the expected format, in Sentence format.

We will be making use of the below python code to convert the XXX file with each word in a line to a sentence.

```
file=open("XXX")
filecontent=''
for i in file:
i=i.rstrip('\n')
if i =='':
filecontent=filecontent+'\n'
else:
filecontent=filecontent+i+' '
print(filecontent)
writer= open("YYYw+")
```

The above code needs to be used on the following list of files to the following files:

- *test.tags.txt → test.tags1.txt*
- *test.words.txt → test.words1.txt*
- *train.tags.txt → train.tags1.txt*
- *train.words.txt → train.words1.txt*

Exercice 1:

Q1: Give the bash command to transform original data into bitexts, the valid set should be an extract of 1000 lines from the train set. (1pt)

We will be making use of the below command in the command prompt of Windows or Terminal of Mac to produce the vocab files, which is the bitext files.

```
onmt-build-vocab --size 1000 --save_vocab src-vocab_valid.txt train.words1.txt
```

Where:

src-vocab_valid.txt – is the bitexts files/ vocab file

train.words1.txt -is the train file converted into sentence data.

Q2: Then, extract three vocabularies from train set using onmt-build-vocab use the char_tokenization.yml file for the character tokenization(2pt) :

- words from the source part of the train set

To convert the words from source part of the train set into vocab file we will be running the below code in the terminal.

```
onmt-build-vocab --size 0 --save_vocab src-vocab.txt train.words1.txt
```

Where:

src-vocab.txt – is the vocab file generated.

train.words1.txt – is the source part of train set.

- character from the source part of the train set

To convert the characters from source part of the train set into vocab file we will be running the below code in the terminal, by making use of the custom tokenizer configuration file, *char_tokenization.yml*

```
onmt-build-vocab --tokenizer_config config/char_tokenization.yml \

--size 0 --save_vocab src-vocab-char.txt train/train.words1.txt
```

Where:

src-vocab-char.txt – is the character vocab file generated.

train.words1.txt – is the source part of train set.

char_tokenization.yml- is the custom tokenizer configuration.

- tags from the target part of the train set

To convert the words from target part of the train set into vocab file we will be running the below code in the terminal.

```
onmt-build-vocab --size 0 --save_vocab tgt-vocab.txt train.tags1.txt
```

Where:

tgt-vocab.txt – is the vocab file generated.

train.tags1.txt – is the target part of train set.

Q3: Give details about the amount of train and test data available for your experiments.

We are having 8936 sentences for training the PoS tagger and 2012 sentences for testing the trained model.

Training the DL model

We will be training the model using the custom model present in the code, seq_tagger_updated.py

Exercice 1: Set up the training by set up the config file (given in the additionnal data for OpenNMT) and then launch the training

We will be making use of the configuration in the configuration file config_sample.yml by updating it as below:

```
model_dir: /Users/prashanth/Downloads/tp3_files/result/

data:
  train_features_file: /Users/prashanth/Downloads/tp3_files/train/train.words1.txt
  train_labels_file: /Users/prashanth/Downloads/tp3_files/train/train.tags1.txt
  eval_features_file: /Users/prashanth/Downloads/tp3_files/train/test/test.words1.txt
  eval_labels_file: /Users/prashanth/Downloads/tp3_files/train/test/test.tags1.txt

  # (optional) Models may require additional resource files (e.g. vocabularies).
  source_1_vocabulary: /Users/prashanth/Downloads/tp3_files/src-vocab.txt
  source_2_vocabulary: /Users/prashanth/Downloads/tp3_files/src-vocab-word.txt
  target_vocabulary: /Users/prashanth/Downloads/tp3_files/tgt-vocab.txt
```

Q1: What is your command line needed to launch the training?

We will be making use of the below command to train the model using the custom model and configuration.

```
onmt-main --config config_sample.yml --model seq_tagger_updated.py train
```

Where:

config_sample.yml – is the custom configuration for training

seq_tagger_updated.py – is the custom model trainer

Q2: How long was your training?

We have trained the model until 100000 steps, it ran for around 18 hours.

We have manually stopped the training, since we deem it should be good enough for evaluation with a loss of around 0.01 to 0.79

Evaluate:

Q1: Give the necessary commands to launch the evaluation of the model on the test set.

To evaluate the model using the eval.py , we first need the trained model to predict the tags(labels) for the test sentences (features).

Inorder to predict the labels, we will be making use of the below command to generate a file predicted.txt which has the tags generated for the test file(test.words1.txt) with sentences.

```
onmt-main \
--config config_sample.yml \
--checkpoint_path result/ \
infer --features_file train/test/test.words1.txt \
--predictions_file predicted.txt
```

The above command uses of the trained model by making use of the checkpoint created during the training.

Once the predicted.txt file is generated, we will be making use of the below command to evaluate the model by comparing it with the test.tags1.txt file , which contains the PoS labels of the features test file test.words1.txt :

```
python eval.py predicted.txt train/test/test.tags1.txt
```

Q2: Give evaluation scores given by the script eval.py and explain them.

Upon making use of the eval.py, we will be getting the below output:

TAG	Precision	Recall	F1
(:	98.7	98.7	98.7
):	100.0	96.25	98.09
$:	100.0	100.0	100.0
#:	100.0	100.0	100.0
CC:	100.0	100.0	100.0
CD:	98.64	98.9	98.77
DT:	99.75	99.01	99.38
EX:	95.83	73.02	82.88
FW:	25.0	4.0	6.9
IN:	99.61	98.86	99.23
JJ:	92.65	84.1	88.17
JJR:	94.06	93.6	93.83
JJS:	93.51	94.74	94.12
MD:	98.94	99.79	99.36
NN:	94.02	91.0	92.48
NNP:	85.16	97.8	91.05
NNPS:	51.54	21.9	30.73
NNS:	89.32	95.56	92.33
PDT:	80.0	53.33	64.0
POS:	100.0	99.09	99.54
PRP:	99.88	99.88	99.88
PRP$:	99.76	100.0	99.88
PUNCT:	99.98	99.94	99.96
RB:	93.28	90.28	91.75
RBR:	80.28	90.48	85.07
RBS:	97.96	100.0	98.97
RP:	25.0	75.0	37.5
TO:	100.0	100.0	100.0
UH:	0.0	0.0	0.0
VB:	94.09	94.24	94.16
VBD:	92.97	95.88	94.41
VBG:	84.2	85.26	84.73
VBN:	88.04	86.94	87.49
VBP:	90.35	92.41	91.37
VBZ:	93.21	95.73	94.45
WDT:	89.11	88.24	88.67
WP:	99.09	100.0	99.54
WP$:	25.0	4.35	7.41
WRB:	98.92	73.6	84.4
All:	85.84	83.89	84.86

The column TAG contains the list of TAGs(labels) from the two files provided for evaluation to the eval.py.

The column Precision contains the precision metrics calculated for each Tag present in the two files.

The column Recall contains the recall metrics calculated for each Tag present in the two files.

The F1 Precision contains the f1 metrics calculated for each Tag present in the two files, by making use of the precision and recall metrics calculated earlier.

The All row contains the average of each metrics calculated for all the tags.