Bahria University, Islamabad

Department of Software Engineering

Artificial Intelligence Lab

(Fall-2021)

Teacher: Engr. M Waleed Khan

Student : M Iqrar Ijaz Malik

Enrollment : 01-131182-021

Lab Journal: 14, 15

Task No:	Task Wise Marks		Documentation Marks		Total Marks (20)
	Assigned	Obtained	Assigned	Obtained	
1	15		5		

Comments:

Signature

LAB

Lab 14

Implementing Machine Learning (Part-1) (NLP, Sentiment analysis)

Introduction

Neuro-linguistic programming (NLP) is a pseudoscientific approach to communication, personal development, and psychotherapy created by Richard Bandler and John Grinder in California, United States, in the 1970s. NLP's creators claim there is a connection between neurological processes (neuro-),

language (linguistic) and behavioral patterns learned through experience (programming), and that these can be changed to achieve specific goals in life. Bandler and Grinder also claim that NLP methodology can “model" the skills of exceptional people, allowing anyone to acquire those skills. They claim as well that, often in a single session, NLP can treat problems such as phobias, depression, tic disorders, psychosomatic illnesses, near-sightedness, allergy, the common cold, and learning disorders. NLP has been adopted by some hypnotherapists and by companies that run seminars marketed as leadership training to businesses and government agencies.

Computers use (analyze, understand, generate) natural language:

Text Processing

- Lexical: tokenization, part of speech, head, lemmas
- Parsing and chunking
- Semantic tagging: semantic role, word sense
- Certain expressions: named entities
- Discourse: coreference, discourse segments

Speech Processing

- Phonetic transcription
- Segmentation (punctuations)
- Prosody

Exercise:

- **Load twitter dataset**
- **Remove stopwords**
- **Tokenize**
- **Add POS**

```
import nltk
import pandas as pd
from nltk.tokenize import TweetTokenizer
from nltk.corpus import twitter_samples,stopwords
all_tweets=twitter_samples.strings('positive_tweets.json')+
twitter_samples.strings('negative_tweets.json')
stop_words=set(stopwords.words('english'))
words=[]
tokenizer=TweetTokenizer()
for tweet in all_tweets:
    for word in tokenizer.tokenize(tweet):
        if word not in stop_words and word.isalpha():
            words.append(word)
print("Words",words)
pos_tags=nltk.pos_tag(words)
print("pos_tag",pos_tags)
```

Output:

```
Words ['top', 'engaged', 'members', 'community', 'week', 'Hey', 'James', 'How', 'odd', '
pos_tag [('top', 'JJ'), ('engaged', 'VBD'), ('members', 'NNS'), ('community', 'NN'), ('w
```

Lab 15

Implementing Machine Learning (Vision,Mnist)

Introduction

Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines. Researchers and enthusiasts alike, work on numerous aspects of the field to make amazing things happen. One of many such areas is the domain of Computer Vision.

There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future. Some of them have been listed below:

- LeNet
- AlexNet
- VGGNet
- GoogLeNet
- ResNet
- ZFNet

Exercise:

Implement CNN on Mnist dataset using tensorflow.

Code

```
import numpy as np
import tensorflow as tf
import tensorflow.keras.layers as KL
import tensorflow.keras.models as KM
## Dataset
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) =
mnist.load_data()
x_train, x_test = x_train/255.0, x_test/255.0
x_train, x_test = np.expand_dims(x_train, axis=-1),
np.expand_dims(x_test,
axis=-
1)
## Model
inputs = KL.Input(shape=(28, 28, 1))
c = KL.Conv2D(32, (3, 3), padding="valid",
activation=tf.nn.relu)(inputs)
m = KL.MaxPool2D((2, 2), (2, 2))(c)
d = KL.Dropout(0.5)(m)
c = KL.Conv2D(64, (3, 3), padding="valid",
activation=tf.nn.relu)(d)
m = KL.MaxPool2D((2, 2), (2, 2))(c)
d = KL.Dropout(0.5)(m)
c = KL.Conv2D(128, (3, 3), padding="valid",
activation=tf.nn.relu)(d)
f = KL.Flatten()(c)
outputs = KL.Dense(10, activation=tf.nn.softmax)(f)
model = KM.Model(inputs, outputs)
model.summary()
model.compile(optimizer="adam",
loss="sparse_categorical_crossentropy",
metrics=["accuracy"])
model.fit(x_train, y_train, epochs=5)
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test Loss: {0} - Test Acc:
{1}".format(test_loss, test_acc))
```

Output:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                11530
=================================================================
Total params: 104,202
Trainable params: 104,202
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1875/1875 [==============================] - 44s 23ms/step - loss: 0.2327 - accuracy: 0.9261
Epoch 2/5
1875/1875 [==============================] - 45s 24ms/step - loss: 0.0797 - accuracy: 0.9747
Epoch 3/5
1875/1875 [==============================] - 44s 24ms/step - loss: 0.0643 - accuracy: 0.9803
Epoch 4/5
1875/1875 [==============================] - 44s 24ms/step - loss: 0.0552 - accuracy: 0.9826
Epoch 5/5
1875/1875 [==============================] - 45s 24ms/step - loss: 0.0482 - accuracy: 0.9843
313/313 [==============================] - 2s 7ms/step - loss: 0.0279 - accuracy: 0.9901
Test Loss: 0.027893491089344025 - Test Acc: 0.9901000261306763
```

Conclusion

I completed the tasks given to us and pasted output above.