location-intent-classify-asr-results

June 21, 2022

[1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
import spacy   # import the spacy nlp
import csv
from sklearn.preprocessing import LabelEncoder # for label encoding
from sklearn.svm import SVC
from sklearn.metrics import classification_report #for evaluation

# Change this path to path of task_dataset
base_path = '/home/manju/Desktop/assign/task_data/'

def readInputFile(csv_file):
    data = pd.read_csv(csv_file)
    #wav_loc = data['path']
    transcript = data['transcription'].str.lower()  # convert to lower case
    location = data['location'].str.lower().str.replace(" ", "_")  # convert to␣
 ↪lower case then replace spaces by _
    #object_category = data['object']
    #location = data['location']
    return list(transcript.str.lower()), list(location) #Convert to list type &␣
 ↪return

def readHypothesiedTranscription(hypo_file):
    with open(hypo_file) as file:
        lines = file.readlines()
        lines = [line.rstrip() for line in lines]
    return lines


## encode transcription to vec foramat using spacy package
def encode_sentences(sentences, embedding_dim, nlp):
    # Calculate number of sentences
    n_sentences = len(sentences)
    print('Number of sentences :-',n_sentences)
    X = np.zeros((n_sentences, embedding_dim))
    # Iterate over the sentences
    for idx, sentence in enumerate(sentences):
        # Pass each sentence to the nlp object to create a document
        doc = nlp(sentence)
        # Save the document's .vector attribute to the corresponding row in x
        X[idx, :] = doc.vector
    return X

### to convert string labels to integers
def label_encoding(labels):
    # Calculate the length of labels
    n_labels = len(labels)
    print('Number of labels :',n_labels)
    # instantiate labelencoder object
    le = LabelEncoder()
    y =le.fit_transform(labels)
    #print(y[:100])
    #print('Length of y : ',y.shape)
    return y

def svc_training(X,y):
    # Create a support vector classifier
    clf = SVC(C=1)

    # Fit the classifier using the training data
    clf.fit(X, y)
    return clf

def svc_validation(model,X,y):
    # Predict the labels of the test set
    y_pred = model.predict(X)

    # Count the number of correct predictions
    n_correct = 0
    for i in range(len(y)):
        if y_pred[i] == y[i]:
            n_correct += 1
    print("Predicted {0} correctly out of {1} examples".format(n_correct, len(y)))
    print("Accuracy : ", (n_correct/len(y)) * 100)

def main(base_path):
    train_file = base_path + 'train_data.csv'
    valid_file = base_path + 'valid_data.csv'
    hypo_file = base_path + 'best_asr_hypothesis.csv'

        ################## DataSet preparation ###############
```

```python
    sentences_valid,labels_valid = readInputFile(valid_file)
    sentences_train,labels_train = readInputFile(train_file)
    sentences_hypo = readHypothesiedTranscription(hypo_file)

    ### print unique elements in list ###
    print("Unique location labels in training data: ", set(labels_train))
    print("Unique location labels in validataion data: ", set(labels_valid))

    print("Loading nlp spacy model :")

    # load nlp spacy model
    nlp = spacy.load('en_vectors_web_lg')

    # Calculate the dimensionality of nlp
    embedding_dim = nlp.vocab.vectors_length
    ###print(embedding_dim)

    print("Encoding train and validation sentences using spacy model")
    train_X = encode_sentences(sentences_train, embedding_dim, nlp)
    test_X = encode_sentences(sentences_hypo, embedding_dim, nlp)

    print("Encoding labels to integers using skleran")
    train_y = label_encoding(labels_train)
    test_y = label_encoding(labels_valid)

    ###Intent classification with SVM | Training Step
    # X_train and y_train was given.
    print("Training SVM for Intent classification i.e predicting location using␣
 ↪transcription")
    model = svc_training(train_X,train_y)

    #Validation Step
    print("SVM Prediction and Evaluation step: ")
    print("Using Best ASR hypotesized transcription from confermer-CTC model")
    #svc_validation(model,train_X,train_y)
    svc_validation(model,test_X,test_y)


### Invoking Main function
if __name__ == "__main__":
    main(base_path)
```

```
Unique location labels in training data:  {'bedroom', 'none', 'kitchen',
'washroom'}
Unique location labels in validataion data:  {'none', 'washroom', 'kitchen',
'bedroom'}
Loading nlp spacy model :
```

```
Encoding train and validation sentences using spacy model
Number of sentences :- 11566
Number of sentences :- 3119
Encoding labels to integers using skleran
Number of labels : 11566
Number of labels : 3118
Training SVM for Intent classification i.e predicting location using
transcription
SVM Prediction and Evaluation step:
Using Best ASR hypotesized transcription from confermer-CTC model
Predicted 2745 correctly out of 3118 examples
Accuracy :  88.03720333547146
```