home

May 29, 2020

[ ]:
```
from flask import␣
 ↪Flask,flash,render_template,request,session,flash,redirect,url_for
from werkzeug.utils import secure_filename
import os
import nltk
from nltk.tokenize import sent_tokenize,word_tokenize
import pandas as pd
import numpy as np
from  sklearn.model_selection import train_test_split,cross_val_score
from  sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score,confusion_matrix,␣
 ↪classification_report


UPLOAD_FOLDER = '/home/user/Desktop/PROJECT/static/upload/'
ALLOWED_EXTENSIONS = {'txt'}
app = Flask(__name__)
app.secret_key = "abc"
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER



@app.route('/')
def form():
       return render_template('form.html')

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/',methods = ['POST', 'GET'])
def result():
    if request.method == 'POST':

        if request.form.get('sub_btn')=='Upload':
```

```
            if 'myfile' not in request.files:
                flash('No file part')
                return redirect(request.url)
            f= request.files['myfile']
            if f.filename == '':
                flash('No selected file')
                return redirect(request.url)
            if f and allowed_file(f.filename):
                filename = secure_filename(f.filename)
                f.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
                session['filename']=filename
                flash('File uploaded  successfully','success')
                f_name=open('/home/user/Desktop/PROJECT/static/upload/
↪'+filename,'r',encoding='utf-8')
                data=f_name.read()
                session['data']=data
                return redirect('/')
            else:
                flash('Unsupported File Format, please select a text␣
↪file','error')
                return redirect('/')


        if request.form.get('sub_btn')=='Preview':
            filename=session['filename']
            data=session['data']

            return render_template("form.html",value=data)
        if request.form.get('sub_btn')=='Properties':
            Data=session['data']
            sent_tokenz=sent_tokenize(Data)
            word_tokenz=[word_tokenize(i) for i in sent_tokenz]
            n=len(word_tokenz)
            x="No.of sentences : "+str(n)
            amb=open('/home/user/Desktop/PROJECT/ambigous_list.txt')
            ambigous_corpus=amb.read().split()
            amb_sent=[]
            amb_words=[]
            for i in sent_tokenz:
                if any(amb in i for amb in ambigous_corpus):
                    amb_sent.append(i)

            amb_words=[]
            for i in sent_tokenz:
                for j in word_tokenize(i):
                    if j in ambigous_corpus:
                        amb_words.append(j)
```

```
            session['AMB_SENT']=amb_sent
            session['AMB_WORD']=amb_words
            y="No.of Ambiguous sentences : "+str(len(amb_sent))
            result="------Ambiguous Sentences Are------- "
            for i in amb_sent:
                result=result+'\n'
                result=result+i

            ambwords="_-------Ambiguous Words Are------- "
            for i in amb_words:
                ambwords=ambwords+'\n'
                ambwords=ambwords+i

            return render_template("form.
↪html",value=Data,nsent=x,nasent=y,ambsent=result,words=ambwords)

        if request.form.get('sub_btn')=='Predict Sense':


            Data=session['data']
            sent_tokenz=sent_tokenize(Data)
            word_tokenz=[word_tokenize(i) for i in sent_tokenz]

            amb_sent=session['AMB_SENT']
            amb_words=session['AMB_WORD']

            with  open("amb_sent.txt","w",encoding="utf-8") as amfile:
                amfile.write("\n".join(amb_sent))


            df=pd.read_csv('mal_corpus.
↪csv',names=['sentence','Ambiguous_word','index','Sense'],skiprows=1)
            df_x=df["sentence"]
            df_u=df["Ambiguous_word"]
            df_y=df['index']
            df_v=df['Sense']
            cv = TfidfVectorizer(input="content",encoding="utf-8",norm="l2")
            xtrain_cv=cv.fit_transform(df_x)
            y_train=df_y.astype('int')
            clf = LinearSVC(penalty='l2', loss='squared_hinge', dual=True,␣
↪tol=0.0001, C=1.0, multi_class='ovr', fit_intercept=True,␣
↪intercept_scaling=1, class_weight=None, verbose=0, random_state=None,␣
↪max_iter=1000)
            clf.fit(xtrain_cv,y_train)

            final_result=""
            for sentence in amb_sent:
```

```
                final_result=final_result+"\nAmbiguous  Sentence:"+sentence+"\n"
                sent_words=sentence.strip('.')
                sent_words=sent_words.split()

                for wrd in amb_words:
                    if wrd in sent_words:
                        final_result=final_result+"Ambiguous Word:"+wrd+"\n"


                test_sentence=[sentence]
                test_sentencevector=cv.transform(test_sentence)
                pred_sentence=clf.predict(test_sentencevector)
                str1=str(pred_sentence)
                df_y1=list(df_y)
                if(pred_sentence in df_y1):
                    for i,j in enumerate(df_y1,0):
                        if(j==pred_sentence):
                            final_result=final_result+"Predicted Sense:
↪"+df_v[i]+"\n"
                            break

            return render_template('form.html',value=final_result)


        if request.form.get('sub_btn')=='Accuracy':
            df=pd.read_csv('mal_corpus.
↪csv',names=['sentence','ambiguous_word','label','sense'],skiprows=1)
            df_x=df["sentence"]
            df_y=df['label']

            cv = TfidfVectorizer(input="content",encoding="utf-8",norm="l2")
           ␣
↪x_train,x_test,y_train,y_test=train_test_split(df_x,df_y,test_size=0.
↪23,random_state=11)

            xtrain_cv=cv.fit_transform(x_train)
            xtest_cv=cv.transform(x_test)

            clf = LinearSVC(penalty='l2', loss='squared_hinge', dual=True,␣
↪tol=0.0001, C=1.0, multi_class='ovr', fit_intercept=True,␣
↪intercept_scaling=1, class_weight=None, verbose=0, random_state=None,␣
↪max_iter=1000)

            y_train=y_train.astype('int')
            y_test=y_test.astype('int')

            clf.fit(xtrain_cv,y_train)
```

```
            pred=clf.predict(xtest_cv)

            Accuracy=accuracy_score(y_test,pred)
            Accuracy=Accuracy*100
            acc="Accuracy of the model : "+str(Accuracy)
            return render_template('form.html',value=acc)




if __name__ == '__main__':
    app.run()
```

[ ]: