```
from py2neo import Graph
import nltk.data
from vaderSentiment.vaderSentiment import sentiment as vaderSentiment
import time
import csv
from language_check import language_checker
from lemmatization import lemmatizer
from feature_identification import identifier


#start tracking the execution time
start_time = time.time()

graph = Graph("http://neo4j:nA>R67;od0ex82X6(<x9C]1|f4SYuM:l@10.8.0.1:7474/db/data")
assert graph.neo4j_version == (2, 3, 2)

# Run the query which gets all the reviews of the listing with the given ID and print them all
all_listings=open('listings_AMS.csv','rb')
spamreader = csv.reader(all_listings, delimiter=' ', quotechar='|')
outputFile = open('output_improved_AMS.csv', 'wb')
outputWriter = csv.writer(outputFile)
outputWriter.writerow(['Listing ID', 'Reviewer ID','Review ID', 'Sentence', 'Sentiment
score','Feature: Accuracy', 'Feature: Check-in','Feature: Cleanliness','Feature:
Communication','Feature: Location','Feature: Value'])
b=0
for row in spamreader:
    if b<2500:
        query1= "MATCH (l:Listing {listing_id:"
        query2="})-[rel:HAS_REVIEW]->(r:Review) "

#       query3="RETURN r.date_string AS DATE"
#       query_date = query1 + str(myrow) + query2 + query3
#       date = graph.cypher.execute(query_date)

        query4="RETURN r.id AS ID"
        query5 ="RETURN r.comments AS COMMENTS"
        query6 =" <- [relation:WROTE_REVIEW] - (p:Person) "
        query7 = "RETURN p.id AS PERSON_ID"
        myrow=str(row).replace('[','').replace('\'','').replace(']','')
        query_comments = query1 + str(myrow) + query2 + query5
        comments = graph.cypher.execute(query_comments)


        query_id = query1 + str(myrow) + query2 + query4
        query_person = query1 + str(myrow) + query2 + query6 + query7
        person_id = graph.cypher.execute(query_person)
        id = graph.cypher.execute(query_id)
        b+=1
# Convert the text review to string and run a language detector. If the langage is english
# the script will slipt the sentences and will send each of them to the sentiment detector.
        index = 0
        for review in comments:
            mystring = str(review).replace('-','').replace('COMMENTS','')
            check = language_checker(mystring)
            if (check=='en'):
# Sentence split based on the english rules
                sentence_detector = nltk.data.load('tokenizers/punkt/english.pickle')
                sentences = sentence_detector.tokenize(mystring.strip())

# Vader sentiment detection - detects the sentiment for each sentence
                for sentence in sentences:
                    vs = vaderSentiment(sentence)
# Lemmatize each word in the text  to bring to the basic form
                    lemmatized = lemmatizer(sentence)
                    probabilities = identifier(lemmatized)
```

```
                    toprint = [myrow,
                    str(person_id[index]).replace('-','').replace('PERSON_ID','').replace('\n','')
                    ), str(id[index]).replace('-','').replace('ID','').replace('\n',''),
                    sentence, str(vs), vs*probabilities[0], vs*probabilities[1],
                    vs*probabilities[2], vs*probabilities[3], vs*probabilities[4],
                    vs*probabilities[5]]
                    outputWriter.writerow(toprint)
            index +=1
        print 'Processing ' + str(b) + ' of 11553...'
print("\nDONE! Execution time is %s seconds." % (time.time() - start_time))
```