

A project report on

Sentimental Analysis of a review/post through Named Entity Identification

For

Xhackathon

By

Sanskar Baid

Mohit Agarwal

M. S. Ramaiah Institute of Technology

Table of contents

* Project description …………………….. 3
* Requirements …………………. 4
* Implementation ………………….. 5
* Conclusion & Challenges ………………. 7
* Code ……………………………………. 8
* References …..…………….. 15

Project Description

Sentiment analysis has been popular for identifying sentiments in user reviews and classify user reviews into categories such as happy, sad, angry or more simply positive, negative, neutral. Some of the user reviews are mostly about entities like person, organization, location etc.

This project involves developing a module called NamedEntityIdentification a feature extraction module to retrieve information from a review. This module should retrieve the named entities like organization, location, person, etc. from a review and all the positive and negative words or sentiments about the entity in the review. The data extracted can further be used to perform sentiment analysis of the reviews. The output of the module is the name of the entity identified, list of descriptions or sentiments (positive and negative words) and classify the review as a positive or negative review.

Requirements

Software/Hardware requirements to implement the project:

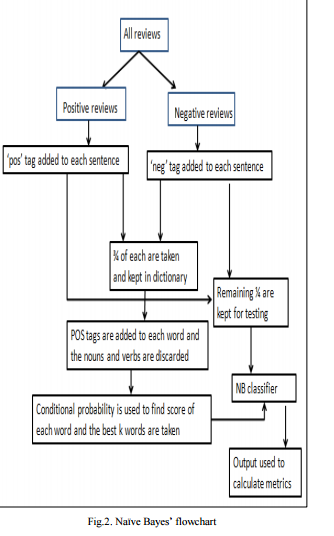
Hardware:

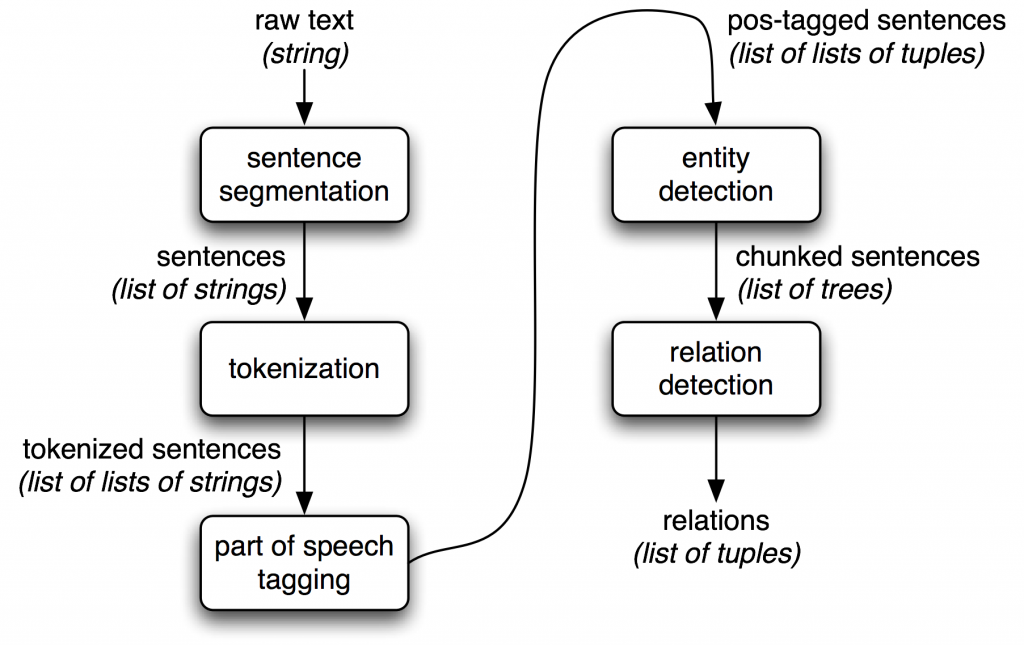
1. Operating System – Windows
2. Minimum 4 GB RAM
3. 500 GB Hard disk

Software:

1. Python
2. NLTK
3. Numpy
4. SciPy

Implementation





## Fig: Information Extraction Architecture

Once we have classified the user’s review, we ask the user if he/she is satisfied by our classification. If they are satisfied, we add this review to our training set, thus increasing the size of the data set and in turn, increasing the accuracy of the result. A user can choose to give multiple reviews if he/she wishes.

Conclusion & Challenges

Conclusion:

Sentiment analysis or opinion mining is a field of study that analyzes people’s sentiments, attitudes, or emotions towards certain entities. This project tackles a fundamental problem of sentiment analysis, sentiment polarity categorization. Online hotel reviews from TripAdvisor.com are selected as data used for this study.

Challenges:

* Installing numpy
* Understanding NLP and NLTK
* Finding proper dataset
* Writing code for training program

Code

**Training Code:**

import os

from nltk.tokenize import word\_tokenize

from nltk.classify.util import accuracy

from nltk.classify import NaiveBayesClassifier

import pickle

def format\_sentence(sentence):

return {word:True for word in word\_tokenize(sentence)}

def save\_classifier(classifier):

f = open('algo.pickle', 'wb')

pickle.dump(classifier, f, -1)

f.close()

train\_pos\_path1 = "trainingData/train/pos\_from\_TripAdvisor"

train\_neg\_path1 = "trainingData/train/neg\_from\_TripAdvisor"

test\_pos\_path1 = "trainingData/test/pos\_from\_TripAdvisor"

test\_neg\_path1 = "trainingData/test/neg\_from\_TripAdvisor"

print('Training model, please wait...')

training\_data = []

for file in os.listdir(train\_pos\_path1):

current = os.path.join(train\_pos\_path1, file)

if os.path.isfile(current):

data = open(current, "r").read()

training\_data.append([format\_sentence(data),'positive'])

for file in os.listdir(train\_neg\_path1):

current = os.path.join(train\_neg\_path1, file)

if os.path.isfile(current):

data = open(current, "r").read()

training\_data.append([format\_sentence(data),'negative'])

print("50% done...")

testing\_data = []

for file in os.listdir(test\_pos\_path1):

current = os.path.join(test\_pos\_path1, file)

if os.path.isfile(current):

data = open(current, "r").read()

testing\_data.append([format\_sentence(data),'positive'])

for file in os.listdir(test\_neg\_path1):

current = os.path.join(test\_neg\_path1, file)

if os.path.isfile(current):

data = open(current, "r").read()

testing\_data.append([format\_sentence(data),'negative'])

print("Completed!!")

bayesModel = NaiveBayesClassifier.train(training\_data)

print('Saving model, please wait...')

save\_classifier(bayesModel)

print('Model Saved! \n')

print('Testing Accuracy, please wait...')

acc =(accuracy(bayesModel,testing\_data) \* 100)

#str(Decimal(acc).quantize(TWOPLACES))

acc = '{0:.2f}'.format(acc)

print('Accuracy: ' + str(acc) + '%')

**Main code:**

import nltk

import os

import pickle

import time

def load\_classifier():

file = open('algo.pickle', 'rb')

classifier = pickle.load(file)

file.close()

return classifier

def format\_sentence(sent):

return {word:True for word in nltk.word\_tokenize(sent)}

def extract\_Enames(sentence): #recursive function to identify entity names

entity\_names = []

if hasattr(sentence, 'label') and sentence.label:

if sentence.label() == 'NE':

entity\_names.append(' '.join([child[0] for child in sentence]))

else:

for child in sentence:

entity\_names.extend(extract\_Enames(child))

return entity\_names

def intersect(a, b):

return (set(a).intersection(b))

train\_pos\_path = "trainingData/train/pos\_from\_TripAdvisor"

train\_neg\_path = "trainingData/train/neg\_from\_TripAdvisor"

test\_pos\_path = "trainingData/test/pos\_from\_TripAdvisor"

test\_neg\_path = "trainingData/test/neg\_from\_TripAdvisor"

pos\_words = "trainingData/Words/Pos"

neg\_words = "trainingData/Words/Neg"

pos=[]

neg=[]

all\_positive=[]

all\_negative=[]

for file in os.listdir(pos\_words):

current = os.path.join(pos\_words, file)

if os.path.isfile(current):

data = open(current, "r").read()

pos.append(format\_sentence(data))

for file in os.listdir(neg\_words):

current = os.path.join(neg\_words, file)

if os.path.isfile(current):

data = open(current, "r").read()

neg.append(format\_sentence(data))

for word in pos:

for key in word:

all\_positive.append(key)

for word in neg:

for key in word:

all\_negative.append(key)

bayesModel = load\_classifier() #Object to call method that loads the classifier file

while(True):

review = input("\nReview: ")

print("")

entity\_names = []

Npos = 0;

Nneg = 0;

positive\_words = []

negative\_words = []

sentences = nltk.sent\_tokenize(review)

all\_words=[]

for sent in sentences: #For each sentence in reveiw

words = nltk.word\_tokenize(sent) #Seperate each word of sentence

all\_words.extend(words)

ptag = nltk.pos\_tag(words) #Give each word a POS Tag

Ename = nltk.ne\_chunk(ptag,binary=True) #Use Chunk to find Entity name

entity\_names.extend(extract\_Enames(Ename))

rev = str(bayesModel.classify(format\_sentence(sent)))

print(rev)

if(rev == "positive"):

positive\_words.append(sent)

Npos = Npos + 1

else:

negative\_words.append(sent)

Nneg = Nneg + 1

var=None

if(Npos > Nneg):

var = "Positive"

print("Given review is Positive.")

else:

if(Nneg > Npos):

var = "Negative"

print("Given review is Negative.")

else:

var = "Neutral"

print("Given review is Neutral.")

if(len(entity\_names) > 0):

result = "Entities are: " + "".join(str(word)+', ' for word in set(entity\_names))

new\_result = list(result)

new\_result[len(result) - 2 ] = "."

print("".join(new\_result))

if(var == "Positive"):

# print("Positive words are: ",positive\_words)

print("Positive words are: ",intersect(all\_positive, all\_words))

if(var == "Negative"):

#print("Negative words are: ",positive\_words)

print("Negative words are: ",intersect(all\_negative, all\_words))

if(var == "Neutral"):

#print("Negative words are: ",positive\_words)

print("Negative words are: ",intersect(all\_negative, all\_words))

# print("Positive words are: ",positive\_words)

print("Positive words are: ",intersect(all\_positive, all\_words))

else:

print('No entity found!!')

valid = input("\nDo you agree with our review? (y/n): ")

if valid == 'y' or valid == 'Y': #Create a new file for trainig set

print("This review has been added to our Training set!")

moment=time.strftime("%Y-%b-%d\_\_%H\_%M\_%S",time.localtime())

if str(var)== 'Positive' :

newF = open(os.path.join(train\_pos\_path,'output'+moment+'.txt'), 'w')

newF.write(review+'\n')

newF.close()

if str(var) == 'Negative':

newF = open(os.path.join(train\_neg\_path,'output'+moment+'.txt'), 'w')

newF.write(review+'\n')

newF.close()

answer = input("\nCheck another review(y/n): ")

if(answer == "y" or answer == "Y"):

print("")

else:

print("Thank you for your feedback!")

break

References

* http://xcelerator.ninja/
* http://xcelerator.ninja/2016/09/01/introduction-to-python-2/
* http://www.nltk.org/book
* Stack Overflow
* Google
* https://github.com/NitishShandilya/NaiveBayesClassifier/tree/master/op\_spam\_train