Natural language processing

SENTIMENT ANALYSIS USING WORD2VEC AND BAG OF WORDS MODEL

SUBMITTED BY : ANKUR SHARMA(101510016), CML-1

Definition AND importance

- The process of computationally identifying and categorizing opinions expressed in a piece of text, especially to determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative, or neutral.
- Shifts in sentiment on social media have been shown to correlate with shifts in the stock market. The Obama administration used sentiment analysis to gauge public opinion to policy announcements and campaign messages ahead of 2012 presidential election.
- Contextual Semantic Search(CSS) algorithm takes thousands of messages and a concept (like Price) as input and filters all the messages that closely match with the given concept.

Applications

- social media monitoring: allows us to gain an overview of the wider public opinion behind certain topics
- ability to quickly understand consumer attitudes and react accordingly
- can be used to give your business valuable insights into how people feel about your product brand or service
- used to identify when potential negative threads are emerging online regarding your business, thereby allowing you to be proactive in dealing with it more quickly

Problems faced

- The human language is complex
- Teaching a machine to analyse the various grammatical nuances, cultural variations, slang and misspellings that occur in online mentions is a difficult process
- Teaching a machine to understand how context can affect tone is even more difficult

Consider the following sentence: “My flight’s been delayed. Brilliant!”

Approach

- Dataset used – IMDb movie review dataset(25000 reviews) downloaded from Kaggle.
- Remove Stopwords using NLTK library
- Remove HTML tags using Beautiful Soup library
- Model Employed for learning features – Bag of Words - using sklearn library
- Another method to learn numeric features from textual data – Word2Vec from genism library
- Model Employed for training features – Random Forest – using sklearn library

STEPS involved

Read the dataset : Use pandas library for reading the dataset downloaded from Kaggle.

Data Cleaning and Text pre-processing : It involves various steps such as removal of html tags as well as stopwords.

Create features from textual reviews : learn a vocabulary from dataset provided and use that vocabulary to make a feature matrix

Use Random Forests to fit on training data : The number of trees used to model the training set is equal to 100. This parameter may be changed according to dataset.

Codes

```
import pandas as pd
train = pd.read_csv("D:\labeledTrainData.tsv", header=0, delimiter="\t", quoting=3)
train.shape
train.columns.values
print (train["review"][0])
from bs4 import BeautifulSoup
example1 = BeautifulSoup(train["review"][0])
print (train["review"][0])
print (example1.get_text())
import re
letters_only = re.sub("[^a-zA-Z]",           # The pattern to search for
                      " ",                   # The pattern to replace it with
                      example1.get_text() )  # The text to search

print (letters_only)
lower_case = letters_only.lower()
words = lower_case.split()
import nltk

from nltk.corpus import stopwords
print (stopwords.words("english") )
words = [w for w in words if not w in stopwords.words("english")]
print (words)
```

Codes

```
def review_to_words( raw_review ):
    # Function to convert a raw review to a string of words
    # The input is a single string (a raw movie review), and
    # the output is a single string (a preprocessed movie review)
    #
    # 1. Remove HTML
    review_text = BeautifulSoup(raw_review).get_text()
    #
    # 2. Remove non-letters
    letters_only = re.sub("[^a-zA-Z]", " ", review_text)
    #
    # 3. Convert to lower case, split into individual words
    words = letters_only.lower().split()
    #
    # 4. In Python, searching a set is much faster than searching
    #   a list, so convert the stop words to a set
    stops = set(stopwords.words("english"))
    #
    # 5. Remove stop words
    meaningful_words = [w for w in words if not w in stops]
    #
    # 6. Join the words back into one string separated by space,
    # and return the result.
    return( " ".join( meaningful_words ))

clean_review = review_to_words( train["review"][0] )
print (clean_review)
num_reviews = train["review"].size
clean_train_reviews = []
```

Codes

```
print ("Creating the bag of words...\n")
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(analyzer = "word",   \
                             tokenizer = None,    \
                             preprocessor = None, \
                             stop_words = None,   \
                             max_features = 5000)

train_data_features = vectorizer.fit_transform(clean_train_reviews)
train_data_features = train_data_features.toarray()

print (train_data_features.shape)
vocab = vectorizer.get_feature_names()
print(vocab)
import numpy as np
dist = np.sum(train_data_features, axis=0)
for tag,count in zip(vocab, dist):
    print (count, tag)
```

Codes

```
print ("Training the random forest...")
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(n_estimators = 100)

#forest = forest.fit( train_data_features, train["sentiment"] )

test = pd.read_csv("D:\labeledTrainData.tsv", header=0, delimiter="\t", quoting=3)

print(test.shape)

num_reviews = len(test["review"])
clean_test_reviews = []
print ("Cleaning and parsing the test set movie reviews...\n")


for i in range(0,num_reviews):
    if( (i+1) % 1000 == 0 ):
        print ("Review %d of %d\n" % (i+1, num_reviews))
    clean_review = review_to_words( test["review"][i] )
    clean_test_reviews.append( clean_review )
```

bag of words model vs word2vec

bag of words	word2vec
• It ignores the context of the word. • It takes very less time to train because semantics of a word are not considered. • It fails when there is sarcasm in the text.	• It does not ignore the context of the word. • It takes a longer time to train because features are learned in vector space according to the context of the word. • It is robust to sarcasm in text.

appendix

- **Stopwords** *- a word that is automatically omitted from a computer-generated concordance or index.*
- ***Bag-of-words -*** *The* ***bag-of-words model*** *is a simplifying representation used in natural language processing and information retrieval(IR).*
- ***Word2Vec -*** *It is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words.*
- **Word embedding** *is the collective name for a set of language modelling and feature learning techniques in natural language processing(NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers.*