Project 4: Report

CSE 535: Information Retrieval (Fall 2016)

Instructor: Rohini K. Srihari,Teaching Assistants: Nikhil Londhe, Lu Meng, Le Fang, Jialiang Jiang

Team Name: Ask Snow

Members:

Pruthvi Mulagala

Vaibhav Sinha (50208769)

Vamsi Krishna

Vinod Veparala

Table of Contents

[Sentiment Analysis using NLTK 3](#_Toc469026033)

[Emotion Tagging using Tensorflow and Alchemy 4](#_Toc469026034)

[Clustering via LDA 6](#_Toc469026035)

[Doc2Vec Model for Tweet Document Vectors 7](#_Toc469026036)

[Clustering Using Doc2vec 8](#_Toc469026037)

[Document Vectors using TF-IDF 8](#_Toc469026038)

[Cluster using KMeans 9](#_Toc469026039)

[Custom Scoring Algorithm 9](#_Toc469026040)

[Overall Topic Summarizing Workflow – 10](#_Toc469026041)

[References 11](#_Toc469026042)

# Sentiment Analysis using NLTK

All the tweets were tagged with sentiments using Sentiment Vader module in NLTK.

4 types of score were generated.

e.g. For the tweet

Cross stitching alllll night #Christmas kill me

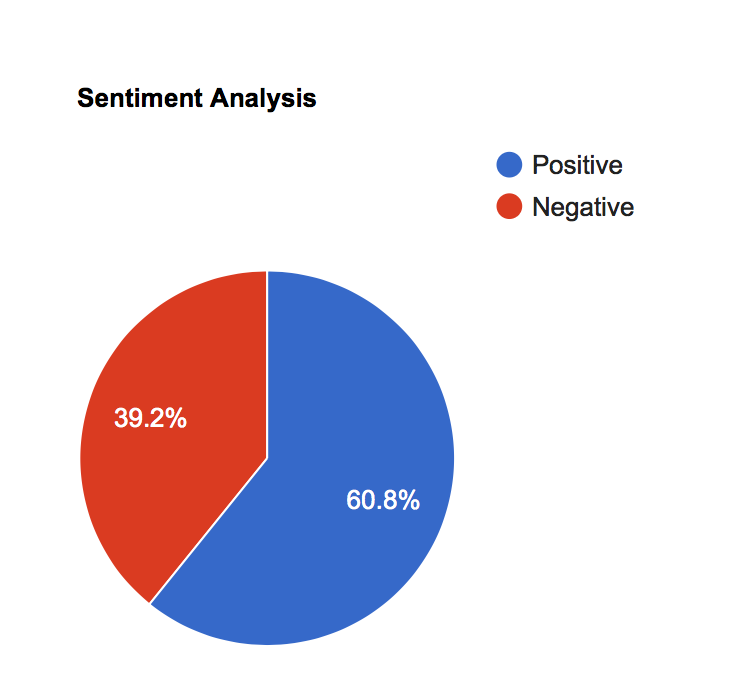
"neg": 0.439, "neu": 0.561, "pos": 0.0, "compound": -0.6908

and for the tweet

I should make a new icon just for #Christmas. Gotta get into that holiday spirit 🎄🎄🎁

{"neg": 0.0, "neu": 0.732, "pos": 0.268, "compound": 0.5267}

For most of the tweets, the result was mostly favoring “compound” sentiment. But since we wanted to extract positive and negative tweets, we took only the values of these two fields.

The results were also shown graphically using a pie chart. A sample is shown below. 

# Emotion Tagging using Tensorflow and Alchemy

Emotions like sadness, anger and joy can also be treated as part of the summary.

In general, we can also model a part of summary as a mixture of emotions conveyed.

Our system is trained to identify emotions like joy, sadness, anger, disgust and fear in tweets.

In order to find out emotions, we used Alchemy API to tag tweets with emotions. But since we had API request limit on Alchemy API, we diligently tagged around 5000 tweets using Alchemy API.

Since, we had around 50,000 tweets in our corpus, we used those 5000 tagged tweets as training data and trained a multilayer artificial neural network to predict the emotion of rest of the tweets.

Tensorflow is used to train a neural network having two hidden layers each having 256 nodes and the output layer having 5 outputs as described above using one hot encoding.

Tweets were converted to feature vectors using Doc2Vec Model.

We partitioned 5000 tweets into two sets, 4000 tweets were used in training and 1000 tweets were used for testing.

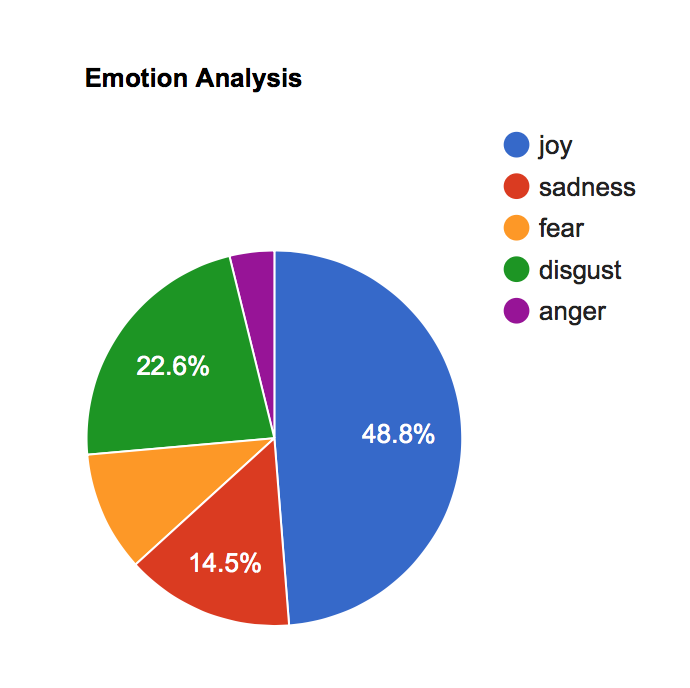
After round 100 iterations, we were able to achieve a training accuracy of more than 90% and around 87% on the testing set.

Since we did not have a large training data, there were not much promising results, but this model can be extended if we had more data.

Plus, most of the data was biased towards joy emotion since Christmas mostly deals with positive tweets.

In Short,

1. Tagged 5000 Tweets using Alchemy API
2. Created Feature Vectors using Doc2Vec
3. Trained a multilayer neural network to tag rest of the 45000 tweets using Tensorflow
4. 90+% accuracy on training set and 87% on testing set



Sample Image which is shown in results which gives general outlook of the tweets in results.

This can help the user get an overall outlook of the people about a particular topic.

[Code for Training Neural Network using Tensorflow](https://github.com/sinha-vaibhav/AskSnow/blob/master/scripts/doc2vecGenerate.py)

# Clustering via LDA

Quoting Wikipedia

“In [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), **latent Dirichlet allocation** (**LDA**) is a [generative statistical model](https://en.wikipedia.org/wiki/Generative_model) that allows sets of observations to be explained by [unobserved](https://en.wikipedia.org/wiki/Latent_variable) groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics.”

We tried to use LDA Model to find sub topics and group similar models together. For this, we used Gensim library in python.

Basically to implement LDA Model, we,

1. Tokenized each tweet using NLTK RegexpTokenizer
2. Stemmed each tweet using Porter Stemmer
3. Removed the stopwords
4. Created a dictionary of all texts using Gensim
5. Created bag of words format for each document using the above dictionary
6. Trained the Gensim model using different number of cluster topics and 10 passes over dictionary

[Code for Model Training](https://github.com/sinha-vaibhav/AskSnow/blob/master/scripts/ldaModel.py)

[Trained Model](https://github.com/sinha-vaibhav/AskSnow/blob/master/ldaModel.lda)

[Topics Identified](https://github.com/sinha-vaibhav/AskSnow/blob/master/lda_topics.txt)

Sample Topic Identified Using LDA

0.058\*"christma" + 0.024\*"need" + 0.020\*"give" + 0.019\*"live" + 0.014\*"away" + 0.013\*"punch" + 0.013\*"pinch" + 0.011\*"hamper" + 0.010\*"hour" + 0.010\*"rt"

To identify which topic a tweet belongs to, lda model gives the probability with which a tweet can be formed from a topic.

From those probabilities, the maximum topic probability was used to identify the corresponding topic of that tweet.

# Doc2Vec Model for Tweet Document Vectors

In order to extract feature vectors for each tweet, Doc2Vec model was used to identify word embedding and generate proper feature vectors for each tweet.

Doc2Vec is an extension of Word2Vec Model from Gensim library. It works on the principle of

Deep learning via the distributed memory and distributed bag of words models from using either hierarchical softmax or negative sampling

Document Vectors were successfully generated.

[Code for training model](https://github.com/sinha-vaibhav/AskSnow/blob/master/scripts/doc2vecModel.py)

[Trained Model](https://github.com/sinha-vaibhav/AskSnow/blob/master/christmas.d2v)

# Clustering Using Doc2vec

Using the trained model, we inferred feature vectors for tweets to generate a feature matrix which is to be sent for KMeans Clustering.

Using the above feature vectors, we used KMeans clustering to cluster similar tweets together.

Unfortunately, the results were not that promising.

The reason can be attributed to the fact that there was not enough data for training Doc2Vec Model and the document vectors were not that accurate.

[Code for Clustering Using Doc2Vec](https://github.com/sinha-vaibhav/AskSnow/blob/master/scripts/cluster_doc2vec.py)

# Document Vectors using TF-IDF

One more method of generating document feature vector was using the Tf-IDF Matrix.

We constructed the term frequency-inverted document frequency matrix using NLTK’s TfidfVectorizer.

We tuned the following parameters for generating document vectors

max\_df=0.99

max\_features=200000

min\_df=0.01

stop\_words='english'

use\_idf=True

ngram\_range=(1,3))

# Cluster using KMeans

The Tf-IDF matrix generated above was sent to KMeans Clustering algorithm to cluster the document into different number of clusters.

On trying, different number of clusters, we narrowed it down to 20 clusters.

Fortunately, clustering using Tf-Idf matrix lead to promising results and hence, we decided to move on with this.

[Code for Tf-IDF Matric Clustering](https://github.com/sinha-vaibhav/AskSnow/blob/master/scripts/cluster.py)

# Custom Scoring Algorithm

In order to find out relevant tweets for summary, we came up with a custom scoring algorithm which added Static Score to all tweets and took the following factors into consideration.

1. Tweet’s Retweet Count – The more the tweet is retweeted, the more chances are that it is relevant for the summary
2. Tweet’s Favorite Count - – The more the tweet is liked by people, the more chances are that it is relevant for the summary
3. Search Scoring using BM25 Model – This makes sure the tweet is relevant for the search query.

Thus, while displaying tweets, tweets were scored by adding the above individual scores.

# Overall Topic Summarizing Workflow –

# References

1. [Natural Language Toolkit](http://www.nltk.org/)
2. [Doc2vec Model](https://radimrehurek.com/gensim/models/doc2vec.html)
3. [LDA Model](https://radimrehurek.com/gensim/models/ldamodel.html)
4. [Tensorflow](https://www.tensorflow.org/)
5. [Sklearn](http://scikit-learn.org/stable/)