Emoticon based Sentiment Analysis of Tweets

Ajinkya Chavan
School of Informatics and Computing
Indiana University Bloomington
Bloomington, Indiana
ajchavan@iu.edu

Ajit Balaga
School of Informatics and Computing
Indiana University Bloomington
Bloomington, Indiana
abalaga@iu.edu

Harkirat Singh
School of Informatics and Computing
Indiana University Bloomington
Bloomington, Indiana
hs30@iu.edu

ABSTRACT

Emoticons are a widely used tool to express simple/complex emotions in the form of a picture and using them is the norm on most online websites/forums, even more so in places like Twitter, where the number of words expressible by an individual are limited. Sentiment analysis is a powerful tool which allows us to describe a user's emotional state based on the words chosen by them. A clear technique for detecting emotions in text with emoticons is not present. Also, in sentiment analysis with words, there often exists a table/database which contains the polarity associated with each word and these values are used to decide the overall sentiment of a sentence/tweet, but no such table exists for emoticons. We experiment with two algorithm techniques to perform sentiment analysis on sentences/tweets filled with one or many emoticons and establish that fuzzy c-means is better than k-means in this regard. We also try to build a table for emoticons to help generate accurate polarities for these emoticons from the sentence/tweets themselves.

1 INTRODUCTION

Social media sites such as Twitter allow users to post short and informal messages, broadcast their opinions on a wide variety of topics and express their emotions. Users express their sentiments on the political environment, religious beliefs, consumer products and personal affairs in a few words. Twitter is a rich resource from which you can gain insights by performing sentiment analysis.

Sentiment analysis is important as it has many real-world applications. Corporate organizations want to uncover insights for better customer management. They want to retain old customers and attract new ones. Sentiment analysis allows companies to perform market research to evaluate customer feedback, without having to send out questionnaires or surveys. Election parties want to study the shift in public opinion about their candidates.

However, any standard algorithm may fail to catch true sentiments hidden in the textual part of tweets. There is the challenge of detection of sarcasm, humor or irony in the texts and it can lead to misclassification of tweets. With different contexts they can mean different things. Misspelled words and grammatical errors can add to the noise in the data set. Tweets containing a mix of positive and negative words can be misclassified as neutral sentiment. Hence, identification of real emotions based on only textual part of tweets is not sufficient as it is very important to understand the actual intent of the author of the tweet.

Sentiment analysis, also known as Opinion mining, can be improved much further by using emoticons. Emoticons were added to social media sites to represent facial features of an author and to spice up emotional cues to text messages. Many complex ideas can be conveyed through simple emoticons. Whereas traditional sentiment analysis determines whether a text is positive or negative (polarity), an extra layer of emoticon analysis can help in classifying messages into further categories like love, joy, surprise, anger, sadness, fear, etc. Emoticons can also help in better classification in case of sarcastic texts.

This study intends to investigate the sentiments of tweets using emoticons. By performing emoticon analysis, this study fills a gap between the area where most research has been focused only on the textual part of tweets.

2 BACKGROUND AND RELATED WORK

Anirban Sen , Koustav Rudrat and Saptarshi Ghosh [7] proposed Natural Language Processing (NLP) techniques to address this challenge and extract low-level syntactic features from the text of tweets, such as the presence of specific types of words and parts-of-speech, to develop a classifier to distinguish between tweets which contribute to situational awareness and tweets which do not. Automatically differentiating such tweets from those that reflect opinion or sentiment is a non-trivial challenge, mostly because of the very small size of tweets and the informal way in which tweets are written, with a lot of emoticons, abbreviations, and so on. Experiments over tweets related to four diverse disaster events show that the proposed features identify situational awareness tweets with significantly higher accuracy than classifiers based on standard bag-of-words models alone.

Soumi Ghosh and Sanjay Kumar Dubey [2] proposed a comparative analysis of Fuzzy C-Means vs K-Means on the Iris dataset. They perform a time complexity comparison between the algorithms for small number of features. The results show that FCM works better than K-means for small number of features.

Li *et al.* [4] present a comparative analysis of K-means and Soft K-Means(Fuzzy) on the BIRCH and Wine dataset from UCI, which contains 100 and 3 clusters respectively. The results demonstrated the effectiveness of the new algorithm in producing consistent clusters and predicting the correct number of clusters in different data sets, even with overlap.

By making use of clustering techniques such as k-means and fuzzy c-means, we attempt to fill the gap in emoticon-based sentiment analysis and develop an algorithm to perform an accurate analysis of a given tweet.

3 DATA COLLECTION AND PRE-PROCESSING

Using a python library (tweepy), we collected the most recent 931240 tweets from the twitter stream, filtering the tweets based on the language, the emoticons present and whether they were retweets. Twitter exposes a stream API to collect such data with filter parameters such as *lang* and *query*. We collected tweets with about 20 selected emojis. From these we removed all tweets which are retweets or were repeated. We then removed all tweets with less than 5 words to ensure an accurate polarity score. We were left with about 156527 tweets matching the above specifications. In these only 6 emojis were present, in numbers exceeding a threshold of about 2000 tweets. Thus, we decided to reduce the number of emojis to 6 and remove all tweets with other emojis. A total of 16219 tweets were left, where each emoji has about 3000 tweets each with the exception of love, which had about 2400 tweets.
Hyperlinks, usernames and stop-words were removed from each tweet, and all the characters were converted to lowercase. Hashtags and emoticons in each tweet were separated. If hashtags were found in camel cased form, each word will be treated as a separate hashtag.

In addition to the above data, we used polarity scores from the website [8] which contained a list of 8221 positive and negative words. This was to check our tweets for accuracy. The stop-words in each tweet were detected by making use of the Stanford NLP website [5], which has about 257 stop-words. We also use the Snowball stemmer from nltk in case of verbs, to look for positive or negative words.

4 EMOTICON-BASED METHOD

Previous studies which focused their experiments on emoticons mainly distributed the intensity of an emotion as an integer polarity. We select 6 of the most commonly used emojis from a list of 751 emojis with respect to their frequency and distinction in the emoji scores[3]. These researched scores are the basis of our approach. The scores are mentioned below.

Emoji Sentiment Ranking	
Emoji	Score
😂	0.221
😜	0.445
❤️	0.746
😭	-0.093
😡	-0.173
😕	-0.397

5 SENTIMENT SCORE

5.1 Scoring parameters and Logic

(1) Since soft-clustering doesn't need extensive training and works on-the-go, we had to ensure that we cover all features of tweets/text. Apart from the general text pre-processing and considering emojis and hashtags together, we are focusing on patterns in parts-of-speech(POS), rather than merely judging on the basis of the tag. We check for noun-adjective pairs, and stemmed verbs for helpful understanding of the potential subjectivity of the tweet.
(2) Since all noisy parts of data have been removed in pre-processing, we are left with emoticons, hashtags and list of keywords in each tweet
(3) We had to decide on the amount of data to be shifted for positive or negative words encountered. Rather than using integer polarity to express intensity of words, we use emoticon scores.
Since we are using **Emoji Sentiment Ranking**[3] as our basis, we take average of all the emoji scores to get **0.124833**. We refer to this value as *averageChangeInSentiment*. This will be the value used for incrementing or decrementing sentiment score

5.2 Scoring components

(1) **Emoji** - For each emoticon, assign the sentiment score corresponding to its value in the emoji ranking site[3]
(2) **Hashtags** - If a hashtag is encountered, if its a positive word, increment by 2**averageChangeInSentiment* and by -2**averageChangeInSentiment* for a negative word.
(3) **POS tags(nltk POS tagger)** -
   (a) If a noun is encountered - if its preceded by an adjective, then we increment or decrement by 2**averageChangeInSentiment* depending on polarity of the word, else we add or subtract the value of *averageChangeInSentiment*
   (b) If a verb is encountered, stem it and check for positive or negative polarity and assign 2**averageChangeInSentiment*. Repeat the same for adjective without the stemming.

6 K-MEANS

Algorithmic steps for K-Means[1] are as follows:

(1) Centroids of clusters are chosen from randomly.
(2) Distances between data points and cluster centroids are calculated.
(3) Each data point is assigned to the cluster whose centroid is closest to it.
(4) Cluster centroids are updated by using the formula:

$$V_i = \sum_{i=1}^{n_i} \frac{X_{ij}}{n_i} \tag{1}$$

where V is set of centers of clusters, X is the dataset, 1<=i<=c; c - number of clusters
(5) Distances from the updated cluster centroids are recalculated.
(6) If no data point is assigned to a new cluster the run of algorithm is stopped, otherwise the steps from 3 to 5 are repeated for probable movements of data points between the clusters.

In our experiment, we are using KMeans from sklearn package in python.

7 FUZZY C-MEANS (FCM)

FCM is an unsupervised clustering algorithm that is applied to wide range of problems connected with feature analysis, clustering and classifier design. This algorithm is used for analysis based on distance between various input data emoticons. The clusters are formed according to the distance between data emoticons and the cluster centers are formed for each cluster.
In fact, **FCM** is a data clustering technique in which a data set is grouped into n clusters with every data emoticons in the dataset related to every cluster and it will have a high degree of belonging (connection) to that cluster and another data emoticons that lies far away from the center of a cluster which will have a low degree of belonging to that cluster.

Algorithmic steps for Fuzzy C-Means [2] are as follows:

(1) Initialize $U = [u_{ij}]$ matrix, $U^{(0)}$
(2) At k-step: calculate the centroid vectors $C^{(k)} = [c_j]$ with $U^{(k)}$
where c is the number of clusters, j is the centroid iteration number, k is the data iteration number
(3) Update $U^{(k)}, U^{(k+1)}$

$$u_ij = \frac{1}{\sum_{k=1}^{C}(\frac{|x_i - c_j|}{|x_i - c_k|})(\frac{2}{m-1})} \tag{2}$$

where m is the membership function(For simplicity, we are keeping m=2)
(4) If $|U^{(k+1)} - U^{(k)}| < \epsilon$ then STOP; otherwise return to step 2. Here ϵ is the termination criterion i.e. the minimum difference between the centroids of each cluster. In our case, ϵ is 0.15.

8 RESULTS

Fuzzy C-Means is also referred to as soft K-Means[6]. In skfuzzy, Fuzzy K-Means is equal to FCM for 1-dimensional data. We take the average of centroids over 10 runs for all the cases to get the following centroids table for our FCM code vs skfuzzy FCM and kmeans.

Centroids			
Emoticon	Our-code	skfuzzy	kmeans
happy	0.2014	0.2105	0.4504
playful	0.4706	0.4515	0.7829
love	0.9443	0.7732	2.0397
sad	−0.1191	−0.0806	0.1922
angry	−0.3858	−0.1884	−0.1393
confused	−0.6466	−0.4485	−0.4719

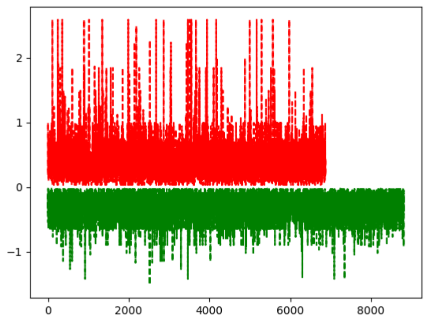

Figure 1: No of iterations vs Sentiment Score(positive-red, negative-green)

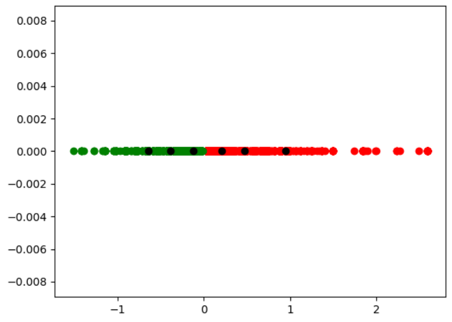

Figure 2: Data distribution with final cluster centroids - Self-coded FCM

Figure 1 shows the distribution of positive and negative tweets, which gives a better idea about the variance between the tweet sentiment scores. For figures 2,3, and 4, the black dots from left to right represent the centroids for each emoticon in the order - **confused, angry, sad, happy, playful and love**. The figures show the variation in the final centroids obtained by using the corresponding technique. We see that our FCM works almost as well as skfuzzy FCM, while K-means doesn't perform as well as the other 2 methods. The accuracy table shows the maximum frequency of accuracy over 50 runs of each code.
In FCM vs skfuzzy FCM, we can see from the centroids table that in case of FCM, it almost always over-estimates the target with the exception of *happy*. On the other hand, K-Means is very close to desired prediction for *angry* and *confused*, but performs horrendously for others, leading to a bad accuracy.

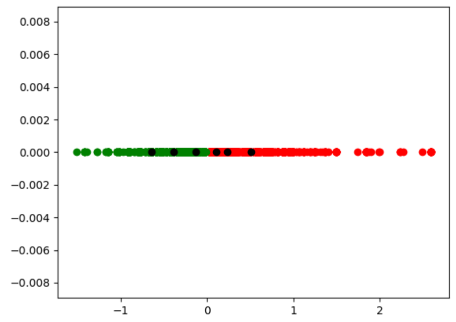

Figure 3: Data distribution with final cluster centroids - sk-fuzzy FCM

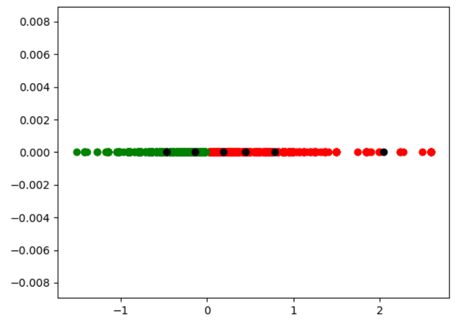

Figure 4: Data distribution with final cluster centroids - K-means

Accuracy			
	Our-code	skfuzzy	kmeans
Accuracy	99.287	100.0	76.076

9 CONCLUSION

We are getting 100% accuracy on maximum occasions for FCM with skfuzzy and around 99.25% for our code. That by no means, concludes that we have solved the Sentiment Analysis problem. There are multiple reasons for this result:

(1) **Heavy text pre-processing** - Out of around a million tweets, we only chose the ones which had the emojis with clear distinction on the Emoji ranking website
(2) **Emoticons** - All the tweets included the emoticons, so it is bound to converge to all the clusters. Also, we couldn't impose heavy penalty on positive or negative words or that would change the whole context of sentences.
(3) **Soft vs Hard Clustering** - With the same data, K-means gives a 76% accuracy, while soft-clustering algorithms handle the boundary conditions properly and have more distinct yet accurate cluster centers.

Considering all the factors, we are assuming emoticons to be the real contributors to the overall tone of the tweet. In real life, however, that isn't always the case. We relied on the objectivity of the tweets. If we could add subjectivity pre-processing before feeding it to clustering algorithms, it is possible to use these techniques on a much more diverse dataset and get more realistic results.
Soft-clustering(Fuzzy) algorithms can be very effective if proper structured dataset is fed to them, and in many cases, they can be faster than Machine Learning techniques like Neural Networks etc. without the pains of tuning required.

10 TEAM MEMBERS CONTRIBUTION

(1) **Ajit Balaga**
  (a) Worked on data collection. Wrote stream_twitter_emoticons.py
  (b) created the dataset of 16219 from around a million tweets
  (c) Wrote Introduction
  (d) Created tables in latex
  (e) Wrote kmeans.py
  (f) Proof-reading
(2) **Harkirat Singh**
  (a) Worked on data pre-processing
  (b) Wrote initial half of sentimentAnalysis.py to create cleaned data for sentiment scoring
  (c) Wrote abstract
  (d) Found and wrote ACM citations
  (e) Wrote plot_1d.py which is used to plot Figure 1 to 4
(3) **Ajinkya Chavan**
  (a) Wrote second half of sentimentAnalysis.py i.e. Fuzzy C-Means code
  (b) Researched related work
  (c) Wrote sections 4 to 8
  (d) Wrote skfuzzy_cmeans.py
  (e) Wrote Conclusion

REFERENCES

[1] Zeynel Cebeci and Figen Yildiz. 2015. Comparison of K-Means and Fuzzy C-Means Algorithms on Different Cluster Structures.
[2] Soumi Ghosh and Sanjay Kumar Dubey. [n. d.]. Comparative Analysis of K-Means and Fuzzy C- Means Algorithms. ([n. d.]).
[3] Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. 2015. Sentiment of emojis. *PLoS ONE* 10, 12 (2015), e0144296. http://dx.doi.org/10.1371/journal.pone.0144296
[4] M. J. Li, M. K. Ng, Y. m. Cheung, and J. Z. Huang. 2008. Agglomerative Fuzzy K-Means Clustering Algorithm with Selection of Number of Clusters. *IEEE Transactions on Knowledge and Data Engineering* 20, 11 (Nov 2008), 1519–1534. https://doi.org/10.1109/TKDE.2008.88
[5] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010
[6] Susana Nascimento and Pedro Franco. 2009. *Unsupervised Fuzzy Clustering for the Segmentation and Annotation of Upwelling Regions in Sea Surface Temperature Images*. Springer Berlin Heidelberg, Berlin, Heidelberg, 212–226. https://doi.org/10.1007/978-3-642-04747-3_18
[7] A. Sen, K. Rudra, and S. Ghosh. 2015. Extracting situational awareness from microblogs during disaster events. In *2015 7th International Conference on Communication Systems and Networks (COMSNETS)*. 1–6. https://doi.org/10.1109/COMSNETS.2015.7098720
[8] Veselin Stoyanov, Claire Cardie, and Janyce Wiebe. 2005. Multi-perspective Question Answering Using the OpQA Corpus. (2005), 923–930. https://doi.org/10.3115/1220575.1220691