Multi-class emotion classification using sentiment analysis

Abhianshu Singla abhianshu@knights.ucf.edu
Rajat Jain bohrarajat@knights.ucf.edu
Yash Shah yashrshah@knights.ucf.edu

Abstract

Natural language processing is an area of computer science concerned with the interaction between computers and human languages. Sentiment analysis and opinion mining is the field of study in natural language processing that analyzes people's opinions, sentiments, and emotions from text. It is one of the most actively researched areas in natural language processing due to its importance in computer science, management science, social media analytics to name a few. We present a software which classifies the input sentence with a sentiment such as joy, sadness, fear, anger, disgust, shame, and guilt. We modeled different classifiers such as Naïve Bayes classifier, Maximum Entropy, and SVM classifier to achieve this task. Our result is a sentiment analysis model trained using the ISEAR dataset which will return the sentiment of the input sentence. Among the models that we built, SVM had the most optimal performance with an accuracy of 54% for in-built libraries and Naive Bayes was 51% accurate for the classifiers we built on our own. It is evident from the results of our models that multi-class emotion classification remains a challenging task due to informal language in user responses, presence of non-textual information such as emoticons, use of slang words/abbreviations, and lack of pre-furnished labeled corpus for training and improving models to achieve near-perfect accuracy. Nevertheless, our model also works effectively for binary classification with an improved accuracy of 81%.

1 Introduction

Emotion plays an important role in human communication. Emotions can be expressed using facial expressions, textual information, or speech. The widespread form of communication on the Web is in the form of text. Hence, we decided to perform sentiment analysis on textual data. Since human emotion is too broad to be categorized using binary classification - positive and negative, we used the ISEAR dataset to for multi-class classification of user responses. The aim of this project is to classify emotions in text. We use different features of text such as unigrams, bigrams, and trigrams for identifying emotion. For emotion classification, we used classifiers like Naive Bayes, Maximum Entropy and Support Vector Machines which have been discussed in detail in section 5. Our model can be applied to other applications such as movie/product reviews, social media analysis based on tweets, which can help provide specific insights and feedback for the merchants. These insights provide vital information upon which business models and strategies can be formulated for effective gains.

2 Background

Research in sentiment analysis has progressed on 2 fronts. First, the complexity of classification models has increased over the years. Linear models such as linear regression and SVM are simple and effective, but advanced concepts such as neural networks have now been used for scalable models. Second, the focus shifted from binary classification of text (thumbs up - thumbs down reviews, and 5-star scale) to multi-class classification. Our project is an extension along these lines where we focus on classifying emotions that provide a specific and detailed response than the positive-negative-neutral classification. Specifically, we wanted to see if we could classify user responses into 7 fine-grained emotion buckets.

Pang et al. [1] proposed the use of machine learning to calculate the sentiment polarity of movie reviews. This polarity was used to classify the movie reviews as positive or negative. They employed Naive Bayes, maximum entropy and support vector machines for emotion classification with maximum accuracy of 82.7% with SVM. Our model achieved 80.5% accuracy on the same dataset for movie reviews using SVM. Boia et al. [2] and Manuel et al. [3] proposed using emoticons and slang expressions to assign a sentiment score to online texts. Bouazizi et al. [4] mention the different features like unigrams, bigrams and trigrams that can be extracted from text to perform sentiment analysis.

Thomas et al. [5] provide the bag-of-words approach using which we can compute the weighted log-likelihood score (WLLS) which can be used to identify and classify emotions into one of the 7 classes. WLLS gives a high score to n-grams associated with a specific emotion and a lower score for n-grams spread out over multiple emotion classes. Using this approach, our SVM model achieved an accuracy of 54%.

3 Data source and preparation

The ISEAR dataset is a freely available dataset with 7666 sentences collected from users over 30 countries. Each sentence is tagged with one of the 7 major emotions - joy, sadness, anger, guilt, fear, shame, and disgust. The dataset had to be cleaned and pre-processed before fitting it to our model. The dataset contains useless statements like ‘no response’, ‘none’, ‘nothing’, ‘don’t remember’, ‘can’t think of anything’, ‘can’t think now’, ‘doesn’t apply’, ‘cannot recall’, etc. which were removed manually. In addition to this, there were certain responses like ‘same as above’, ‘already explained before’, etc. which referenced to the statement users made at earlier stages. We replaced these with the statements that they are referring to and its associated emotion. The next stage of pre-processing involved dealing with apostrophes and returning words such as ‘i’ve’,’i’m’,’haven’t’ and replacing them with their true forms ‘i have’,’i am’, and ‘have not’. After this initial cleaning process, the dataset was reduced to 7533 sentences.

4 External software

We have used the following Python packages for building our model. We have used Natural Language Toolkit or NLTK for data processing tasks such as tokenization,stop words removal, and stemming. Numpy has been used for scientific computing specifically, the numpy array has been used for WLLS calculations for emotion classification. Pandas has been used for data analysis while importing the dataset. Scikit-learn has been used to import standard library functions available for Naive Bayes, Maximum Entropy and SVM to compare the performance of our model. tkinter has been used for designing the GUI which handles user input and displays appropriate output.

5 System Implementation

Our software has 5 components: Data gathering and cleaning, data processing, feature extraction, sentiment classification, and user interface.

Data gathering and cleaning: Data is acquired from the dataset in the form of sentences and their emotion tags. Data is cleaned by removing invalid and/or blank responses to ensure the data being used by our model is correct and usable.

Data processing: We split the data into training and test data. We use 80% of the data as the train set and the remaining 20% serves as the test set upon which we will evaluate our model's performance. In the training set, each sentence was split into set of tokens. Each token was converted to lowercase in order to eliminate ambiguity with uppercase words. Stop words such as 'when','a','an',etc. and punctuation marks such as '.','?',!',etc. were removed to reduce the sentence into a set of meaningful tokens which were closely related to the emotion associated with it. For a given sentence, following figure shows the various stages of data processing that we performed.

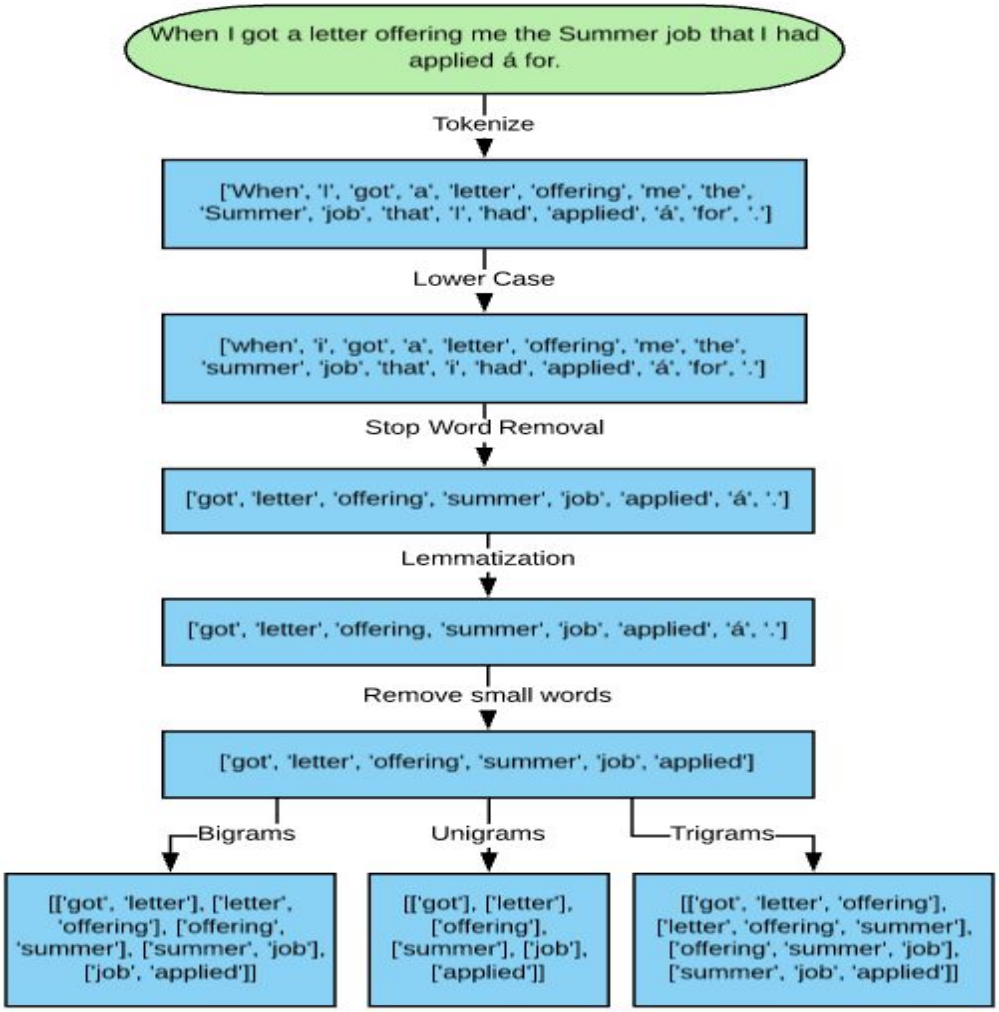

fig 1: Data processing procedure

Feature extraction: We extract unigrams, bigrams, and trigrams from individual sentences. These are features useful for emotion identification. Unigrams are nothing but individual tokens. We use unigrams to calculate the probability of a given word belonging to a particular emotion class. Bigrams are a pair of adjacent unigrams. The bigrams within a sentence are all possible word pairs formed from neighboring words in a sentence.Trigrams are similar to bigrams except that they are 3-word pairs. These n-grams are important as they help improve prediction accuracy. It is easier to predict an emotion for a set of words in comparison to predictions based on a single word. The more data we have, the quality of a model's predictions improves.

Sentiment classification: We create a bag of words which consists of unigrams, pair of unigrams-bigrams and pair of unigrams-bigrams-trigrams. We create a feature vector space using this bag of words. We calculate the weighted log-likelihood score (WLLS) for each point in this vector. This WLLS score is used for feature selection.We select the features with WLLS scores above a certain threshold. This threshold has been manually selected to 0.00099 to limit the number of features being selected. The formula for calculating WLLS is:

$$WLLS(w_i, c_j) = P(w_i \mid c_j) * \log (P(w_i \mid c_j) / P(w_i \mid \neg c_j))$$

where w_i represents the n-gram (unigram, bigram and trigram) whose score is to be evaluated and c_j is the emotion class with respect to whom the score is being evaluated.
$P(w_i \mid c_j)$ is the ratio of the count of n-gram w_i in the emotion class c_j to the count of all the words in that class.
$P(w_i \mid \neg c_j)$ is the ratio of the n-gram w_i in class $\neg c_j$ (all classes except c) to the count of all other words in the class.
We have used 3 classifiers for sentiment classification:Naive Bayes, Maximum Entropy, and Support Vector Machines.

a. Naive Bayes:

In this approach, we assign a sentence 'd' a class 'c' with the highest probability for P(c|d). For this approach, we follow Bayes rule.

$$P(c \mid d) = (P(c)*P(d \mid c)) / P(d)$$

Naive Bayes classifier uses this rule, assuming the frequencies are conditionally independent, and we use the following formula: $P_{NB}(c|d) = (P(c) * P(f_i \mid c)^{ni(d)}) / P(d)$, where f_i belongs to the set of features.
The basic idea is to get the probability of a particular word belonging to a particular class among all the other words belonging to that class.
In this approach, we have created an array with 7 rows, one for each class, and n columns (n = number of features). We insert the respective probabilities for each feature (unigrams, bigrams, trigrams, unigrams-bigrams, unigrams-bigrams-trigrams) to its corresponding class. For prediction, we multiply the entries of this table with its feature vector table entries and return the index (class) at which we receive maximum value.

b. Maximum Entropy:

Maximum Entropy is another technique which has proven to be effective in natural language processing applications. The formula for M.E is as follows:

$$P_{ME}(c \mid d) = (1 / Z(d)) * \exp (\Sigma \lambda i, c \, F_{i,c}(c,d))$$

Maximum Entropy does not assume the relation between its features and performs better even when conditional independence is not met. $\lambda i, c$ are feature-weight parameters. A strong $\lambda i, c$ indicated a strong likelihood of the feature belonging to that emotion class. Z(d) is a normalization function $F_{i,c}$ is a feature function which has a value of 1 if $n_i(d) > 0$ and c' = c.

The basic idea is to see which distribution can represent the available data in the best possible manner. We achieve this by using the normalization function and exponential formula stated above. You need to find a value $P_{ME}(c|d)$ such that it has maximum entropy value to be classified into class c.

c. Support Vector Machines:

Support vector machines are highly effective in classification, outperforming Naive Bayes and Maximum Entropy. The basic idea is to find a hyperplane 'w' that not only distinguishes each class but the distance of the margin or boundaries should be as large as possible. Basically, given a feature space, we have to create 7 different boundaries for each class and map the user responses into their respective classes.

$$w := \sum_j a_j c_j d_j, \text{ where } a_j > 0$$

Given a hyperplane and an observation, SVM will classify which class it belongs to depending upon which side of the hyperplane it lies on.

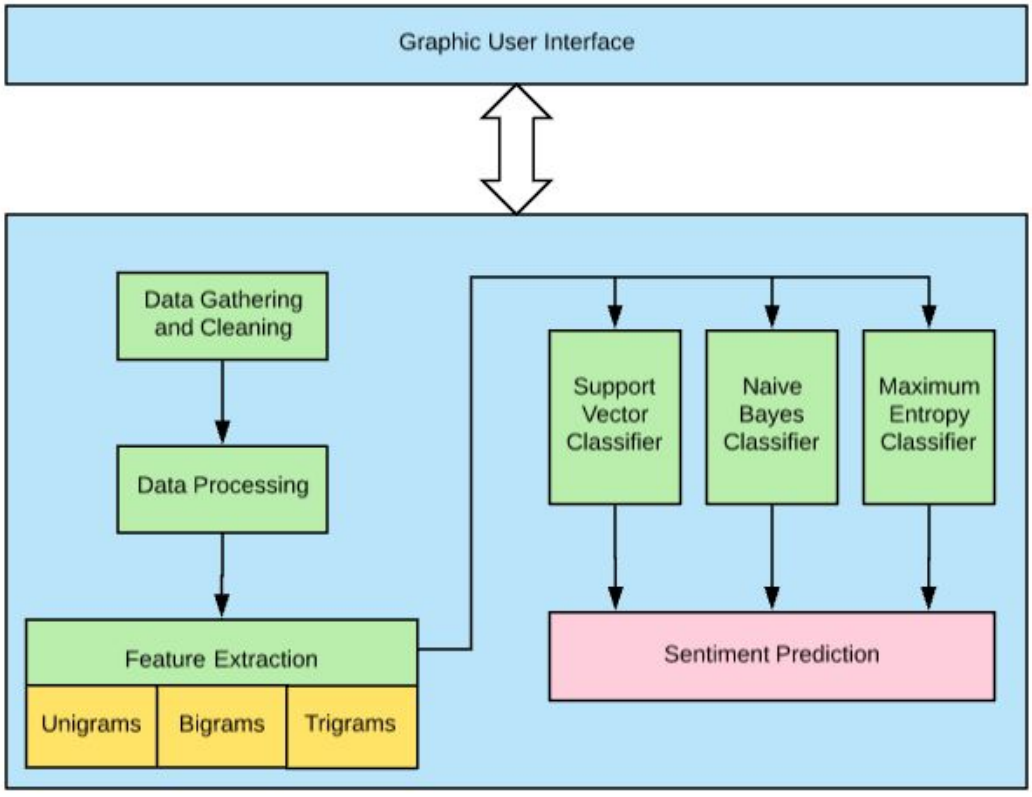

fig 2: System architecture

User interface:

We have used tkinter library to design the user interface which accepts user response and displays the classified emotion as its output to the user.

6 Experiments and results

The test set created during the initial stages is used to evaluate the performance of our model. Across all the papers that we studied during our research, we saw an accuracy range of 55-65% for multi-class classification models.

Features→	Unigrams			Bigrams			Trigrams			Unigram-Bigram			Unigram-Bigram-Trigram		
Epochs ↓	NB	ME	SVM	NB	ME	SVM	NB	ME	SVM	NB	ME	SVM	NB	ME	SVM
1	0.32	0.49	0.49	0.34	0.43	0.49	0.34	0.40	0.46	0.34	0.42	0.47	0.35	0.41	0.43
2	0.29	0.46	0.51	0.33	0.43	0.47	0.32	0.42	0.48	0.31	0.44	0.49	0.32	0.36	0.45
3	0.35	0.47	0.5	0.33	0.43	0.50	0.31	0.38	0.46	0.32	0.45	0.48	0.31	0.42	0.47
4	0.3	0.47	0.51	0.30	0.42	0.49	0.35	0.41	0.48	0.34	0.43	0.51	0.33	0.41	0.47
5	0.32	0.48	0.52	0.31	0.46	0.49	0.34	0.38	0.49	0.33	0.47	0.50	0.34	0.40	0.48
6	0.32	0.46	0.50	0.35	0.40	0.45	0.32	0.40	0.48	0.32	0.48	0.50	0.35	0.36	0.44
7	0.32	0.49	0.53	0.34	0.43	0.49	0.32	0.43	0.49	0.35	0.46	0.48	0.34	0.42	0.46
8	0.32	0.46	0.50	0.31	0.41	0.49	0.33	0.44	0.44	0.31	0.44	0.47	0.36	0.41	0.44
9	0.32	0.49	0.51	0.32	0.43	0.49	0.31	0.40	0.49	0.34	0.45	0.46	0.36	0.42	0.45
10	0.31	0.50	0.52	0.31	0.41	0.49	0.34	0.41	0.48	0.33	0.46	0.51	0.33	0.40	0.46
11	0.31	0.48	0.52	0.33	0.39	0.46	0.33	0.43	0.47	0.31	0.47	0.52	0.32	0.43	0.48
12	0.32	0.49	0.54	0.33	0.41	0.49	0.32	0.41	0.45	0.32	0.44	0.48	0.34	0.41	0.47
13	0.32	0.49	0.54	0.34	0.39	0.50	0.31	0.39	0.46	0.34	0.44	0.46	0.35	0.38	0.44
14	0.34	0.48	0.49	0.33	0.42	0.49	0.34	0.40	0.47	0.35	0.46	0.49	0.31	0.39	0.43
15	0.31	0.48	0.52	0.33	0.43	0.49	0.35	0.42	0.47	0.33	0.48	0.52	0.31	0.41	0.43
16	0.31	0.49	0.53	0.35	0.43	0.51	0.31	0.41	0.46	0.34	0.44	0.51	0.34	0.42	0.45
17	0.34	0.47	0.49	0.35	0.43	0.47	0.32	0.40	0.47	0.32	0.45	0.49	0.33	0.43	0.45
18	0.31	0.46	0.51	0.32	0.41	0.49	0.30	0.41	0.46	0.33	0.44	0.47	0.34	0.41	0.46
19	0.30	0.49	0.52	0.31	0.41	0.48	0.31	0.43	0.48	0.31	0.42	0.48	0.33	0.39	0.47
20	0.32	0.49	0.51	0.32	0.39	0.50	0.31	0.38	0.49	0.34	0.46	0.50	0.32	0.39	0.48

Table 1: Accuracy chart for classifiers using in-built library functions

Above table shows the accuracy chart for each classifier obtained in each feature using in-built libraries. Naive Bayes had a best case accuracy of 36%, Maximum Entropy had an accuracy of 49% whereas SVM had the best performance with 54%.

Using the classifiers that we built from scratch, Naive Bayes obtained an accuracy of 51%, Maximum Entropy had an accuracy of 33% and SVM had an accuracy of 35%. Among all 3 classifiers, SVM performed the best with in-built libraries whereas for classifiers we developed on our own, Naive Bayes performed the best. Among the feature set, unigrams provided the best results for both in-built libraries and our classifiers.

fig 3: User interface

7 Conclusion

We managed to implement 3 classifiers for sentiment classification and compared it with the performance of in-built library functions. Our model performed better for linear models such as Naive Bayes, where its performance was better than in-built functions by 10-15%. Although, the performance of in-built functions was superior for non-probabilistic methods like SVM. The dataset we worked on was relatively small (approximately 7500 sentences), and if our model were to be trained using a corpus larger than the one we had, we believe our model's accuracy would improve further. This is validated by the fact that our model was 81% accurate on binary classification task over 10000 sentences.

Emotions are subjective. For instance, when a person is attacked, that might evoke an angry response from some while it might lead some to be scared. So inherently there becomes a bias favoring the participants involved in the study that created the database. hence, the accuracy of the classification model fluctuates/varies/is affected. For effective multi-class classification, we need a corpus similar to Penn Treebank with labeled emotions, thus ensuring more data and effective training for models.

8 References

[1] Pang, Lee and Vaithyanathan. "Thumbs up? Sentiment Classification using Machine Learning Techniques", 2002.

[2] Marina Boia, Boi Faltings, Claudiu-Cristian Musat, Pearl Pu. "A :) Is Worth a Thousand Words: How People Attach Sentiment to Emoticons and Words in Tweets", 2013.

[3] K. Manuel, K. V. Indukuri, and P. R. Krishna. ''Analyzing Internet slang for sentiment mining'', 2010.

[4] Mondher Bouazizi And Tomoaki Ohtsuki. "A Pattern-Based Approach for Multi-Class Sentiment Analysis in Twitter", 2017.

[5] Bincy Thomas, Vinod P, Dhanya K A. "Multiclass Emotion Extraction from Sentences", 2014.