**Natural Language Processing Project Report**

**Group 10**

Zhiyi Chen, Chih-Han Chi, Ming Yuan Huang, Shih-Ting Weng, Pei-Ling Wu

**1.0 Project Description and Objectives**We aim to utilize the emotions dataset from Kaggle to conduct sentiment analysis and build a multi-classification model. Our objective is to identify positive and negative emotions from texts. After the model is built, we will feed the reviews from Yelp to understand the sentiment of customer’s comments and how the customers feel. This application will help establishments to recognize the reviews that might negatively affect their reputation. Establishments can then take actions to improve their service and avoid harmful reviews in the future.

**2.0 Data Descriptions**

We will retrieve the emotions dataset from Kaggle (Emotions dataset for NLP, <https://www.kaggle.com/datasets/praveengovi/emotions-dataset-for-nlp?select=val.txt>) which has 16,000 trains, 2,000 tests and 2,000 validation.The dataset contains the texts with a label of emotion. This will help us to complete NLP classification tasks and build the machine learning models. For Yelp review data, we will also use the dataset from Kaggle (Yelp Dataset, <https://www.kaggle.com/datasets/yelp-dataset/yelp-dataset>) for model prediction.The Yelp dataset is in Json format and we will mainly use review data from the restaurants to generate outcome.

**3.0 Methodology**

We will do the preprocessing using the ‘re’ package to remove punctuations and do lower casing. We would like to know about the word importance using BoW(Bag of Words) and TF-IDF(Term Frequency-Inverse Document Frequency). After this, we will get the top correlated unigrams, bigrams, trigrams, and wordcloud for each category.

After completing the preprocessing and exploratory data analysis, we will use two methods to train our models, and may also do some stemming and lemmatization to check whether the accuracy gets better.

The first method is directly importing the text data into our models, we will try tuning the parameters to receive the best performance model (manually or by the cv Gridsearch).

The models we will use are as follow:

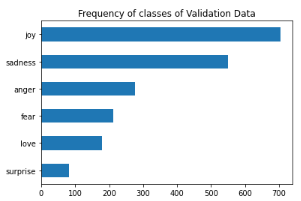
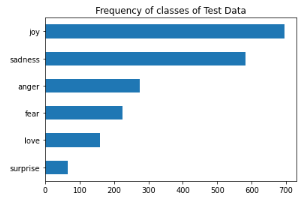
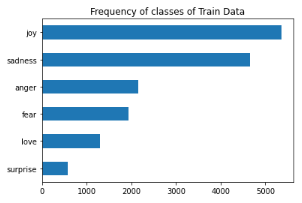
1. Random Forest
2. Linear Support Vector Machine
3. Multinomial Naive Bayes
4. Logistic Regression

**4.0 Key findings**

4.1 Data exploration in emotion dataset

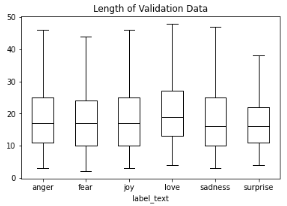
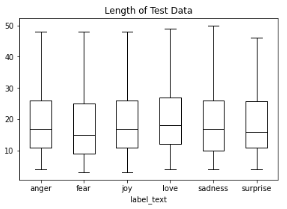
4.1.1 Frequency of classes among train, test and validation

After plotting out the frequency of each class among the train, test and validation dataset, we could obviously see that the data follow a similar distribution of each dataset. In general, pattern shows that the most frequent classes are “joy” and “sadness”, and the rare ones are “love” and “surprise”. Therefore, we might have issues with unbalanced classes in our classification model.

****

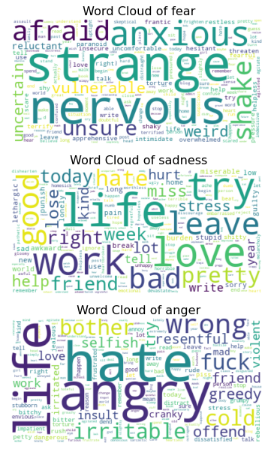
4.1.2 Sentence length of classes among train, test and validation

The average length of the comments is around 19 words, and around range 10 to 25 words across train, test and validation in each class from the box plots. No significant difference among classes.



4.1.3 Word cloud of classes in training set

In order to get further context of each class, it is helpful to plot out word cloud graphs. We preprocess the data with stopwords in the spaCy package, and add customized stopwords such as "ill", "don" , "t", "m", "feel", "make","know", "feeling",'really','go','time','s','ve','think',”will','want','little','still','day','people','need','thing','one','bit','way','come','look','find','start' to exclude more common words. Below shows our word cloud plots.



There is no doubt that most positive words appear in positive classes and vice versa. It is interesting to notice that the word “weird” is a common word showing in the class “surprise” while “strange” is negatively shown in class “fear”. Another point of interest is that the words such as “work”, “life” and “work” are shown in class “sadness”. From the word cloud, we could also feel the strong emotion words such as “love”, “amazed”, “hate”, and “angry” in classes.

4.1.4 Word importance of classes in training set

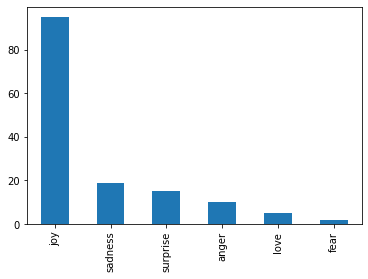
Besides using word clouds, we use CountVectorizer() and TfidfVectorizer() and apply unigrams and bigram to the text to understand the word importance. For most of the cases, we find similarity for both feature extraction, but there is still a slight difference on the result of word importance.

|  | **CountVectorizer(min\_df=0.0001, ngram\_range=(1,1))** | **CountVectorizer(min\_df=0.0001, ngram\_range=(2,2))** | **TfidfVectorizer(min\_df= 0.0001, ngram\_range=(1,1), sublinear\_tf=True))** | **TfidfVectorizer(min\_df= 0.0001, ngram\_range=(2,2), sublinear\_tf=True))** |
| --- | --- | --- | --- | --- |
| **surprise** | funny, surprised, impressed, curious, amazed | wake daze, remember shocked, like admit, daze confuse, pretty impressed | funny, surprised, impressed, curious, amazed | help curious, remember amazed, impressed work, daze confuse, pretty impressed |
| **love** | naughty, loyal, nostalgic, horny, sympathetic | skin lovely, fond memory, love love, like naughty, like care | delicate, naughty, horny, nostalgic, sympathetic | like hot, fond memory, love love, like naughty, like care |
| **joy** | confident, glad, content, satisfied, successful | cool cool, like perfect, like worthwhile, like important, like valuable | rich, brave, glad, successful, content | self assure, like worthwhile, like perfect, like valuable, like important |
| **fear** | unsure, uncertain, reluctant, apprehensive, vulnerable | remember terrified, begin anxious, paranoid like, like paranoid, absolutely terrified | uncertain, unsure, apprehensive, insecure, vulnerable | hesitant post, nervous anxious, begin anxious, paranoid like, like paranoid |
| **sadness** | awkward, lethargic, punish, melancholy, miserable | like worthless, target blank, like fake, like punish, like miss | homesick, awkward, punish, miserable, gloomy | like doom, like worthless, like fake, like punish, like miss |
| **anger** | angry, dangerous, resentful, greedy, irritable | like greedy, like selfish, like rude, like heartless, like bother | irritated, resentful, cranky, irritable, greedy | like stubborn, like greedy, like heartless, like selfish, like bother |

4.2 Data exploration in Yelp data

We self labeled 147 review, and below are the distribution:

| Joy | Sadness | Surprise | Anger | Love | Fear |
| --- | --- | --- | --- | --- | --- |
| 96 | 19 | 15 | 10 | 5 | 2 |



4.3 Model accuracy comparison

After deciding the feature extraction, we have decided to use four different models to train and test, they are random forest, support vector machine and Multinomial Naive bayes, logistic regression. We evaluate the model based on precision, recall and average f1 scores. The results are shown in the table below, the support vector machine model performs slightly better than the others in both cases.

1. CountVectorizer(max\_df=0.95, min\_df=0.0001, ngram\_range=(1, 2))

| Random Forest    Avg. F1 score 0.858 | Linear Support Vector Machine    Avg. F1 score 0.8620 |
| --- | --- |
| Multinomial Naive Bayes    Avg. F1 score 0.8125 | Logistic Regression    Avg. F1 score 0.8585 |

\* 'surprise':1, 'love':2 , 'joy':3 , 'fear': 4, 'sadness': 5, 'anger':6

1. TfidfVectorizer(min\_df=0.0001, ngram\_range=(1, 2), sublinear\_tf = True)

| Random Forest  Avg. F1 score 0.8620 | Linear Support Vector Machine    **Avg. F1 score 0.866** |
| --- | --- |
| Multinomial Naive Bayes    Avg. F1 score 0.7345 | Logistic Regression    Avg. F1 score 0.8515 |

\* 'surprise':1, 'love':2 , 'joy':3 , 'fear': 4, 'sadness': 5, 'anger':6

4.4 Yelp data feeding

1. CountVectorizer(max\_df=0.95, min\_df=0.0001, ngram\_range=(1, 2))

| Random Forest    Avg. F1 score 0.5274 | Linear Support Vector Machine    Avg. F1 score 0.5616 |
| --- | --- |
| Multinomial Naive Bayes    **Avg. F1 score 0.6027** | Logistic Regression    Avg. F1 score 0.5137 |

\* 'surprise':1, 'love':2 , 'joy':3 , 'fear': 4, 'sadness': 5, 'anger':6

1. TfidfVectorizer(min\_df=0.0001, ngram\_range=(1, 2), sublinear\_tf = True)

| Random Forest    Avg. F1 score 0.5685 | Support Vector Machine    Avg. F1 score 0.5891 |
| --- | --- |
| Multinomial Naive Bayes    **Avg. F1 score 0.6233** | Logistic Regression    Avg. F1 score 0.5411 |

\* 'surprise':1, 'love':2 , 'joy':3 , 'fear': 4, 'sadness': 5, 'anger':6

For using the emotion data from kaggle to train, test, and validate the Bag of Words and Term Frequency-Inverse Document Frequency models. By looking at the micro-average F1 score, the best models are Linear SVC model in both Bag of Words and Term Frequency Inverse Document Frequency models. However, after applying our models to Yelp data, we can find out that the best performance model has changed to Multinomial Naive Bayes models in both BoW and TF-IDF models. This shows that there are some overfitting happening in our training models.

Ex. CountVectorizer(max\_df=0.95, min\_df=0.0001, ngram\_range=(1, 2)) for Logistic Regression

**TRAINING**

**Precision [0.99536823 0.99327818 0.99813119 0.99156442 0.99572467 0.9825784 ]**

**Recall [0.99536823 0.9917398 0.99608355 0.99156442 0.99828547 0.98601399]**

**F1 score [0.99536823 0.9925084 0.99710632 0.99156442 0.99700342 0.98429319]**

**Avg. F1 score 0.995375**

**TESTING**

**Precision [0.87822878 0.86425339 0.91641791 0.69879518 0.8836425 0.6835443 ]**

**Recall [0.86545455 0.85267857 0.88345324 0.72955975 0.90189329 0.81818182]**

**F1 score [0.87179487 0.85842697 0.8996337 0.71384615 0.89267462 0.74482759]**

**Avg. F1 score 0.8685**

**Profit: 191.64545795383364**

For evaluation, we choose recall, precision , F1 score and profit as our reference. When we look at the precision and recall, we can see that the model performed better on classifying those true positive

**5.0 Challenges and Learning**

5.1 Challenges

1. Overfitting

The data we used for training, testing, and validating is too clean, this causes our models to have overfitting issues. Hence, we tuned the classifier and imported the Yelp data to validate the model again. However, overfitting happened. Therefore, in the future, we would train the model with more real-world data.

1. Self-label Yelp data

The original data from Yelp restaurant review contains only text without any emotional labels, therefore we self-labeled 147 Yelp restaurant reviews per 6 classes in our model. However, problems such as class consistency might occur due to subject opinion.

1. Insufficient data variety

The data from kaggle is mostly in the form of one sentence with simple structure and short in the number of words in each text. And the text from the Yelp review consists of at least 3 sentences.

5.2 Learning

Traditionally, we use Yelp review data for sentiment analysis to understand positive/neutral/negative reviews. Now, we leap from the routine to further dig on emotions by combining two datasets (emotions and Yelp) and our multi-classification models. This provides more business value for restaurant owners to understand how customers feel about their dining experience.

**6.0 Conclusion**

Online reviews can have a big payoff, however, with large amounts of information, the restaurant was unable to spend much time on reading and processing the problem from the reviews. Therefore, we create this multi-classifier model to determine the satisfaction so that it can be used to help restaurants identify the important reviews and improve the experience of customers. For example, we can identify those comments that need emergency attention such as anger or fear so restaurants can review them in order to make changes and provide solutions, at the same time we can identify the joyful comments so that restaurants can do some retention strategies to encourage customers to come again.

In addition, there is overfitting happening in our model due to the clean data and the lack of data variety. In the future, we would train the data with more data to avoid the overfitting and make it more useful for real word business uses.