Emotion Classification using NLP and Machine Learning

Introduction

This project aims to develop a model that can classify text into different emotion categories. This is a common task in Natural Language Processing (NLP) known as sentiment analysis or emotion detection.

About the Dataset

The dataset used in this project is a collection of documents and their associated emotions, which is highly beneficial for NLP classification tasks. The dataset is split into training, testing, and validation sets for building the machine learning model.

An example of the data is as follows: "i feel like I am still looking at a blank canvas blank pieces of paper" is associated with the emotion 'sadness'.

The dataset was prepared using a technique described in this paper. Special thanks to Elvis and the Hugging Face team https://lnkd.in/eXJ8QVB for providing the dataset.

The dataset serves as an inspiration for the community to develop emotion classification models using an NLP-based approach. The model developed in this project can answer questions based on customer reviews such as:

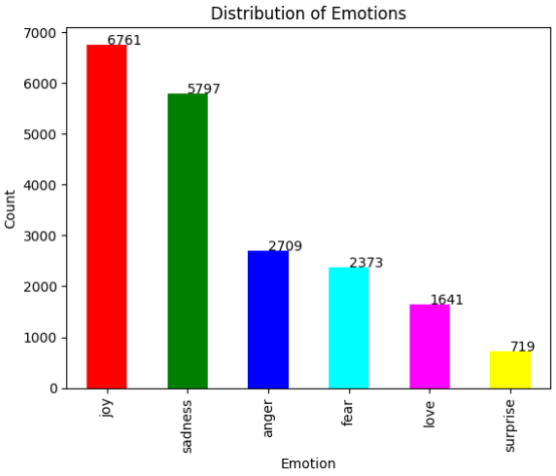
- What is the sentiment of your customer comment?
- What is the mood of today's special food?

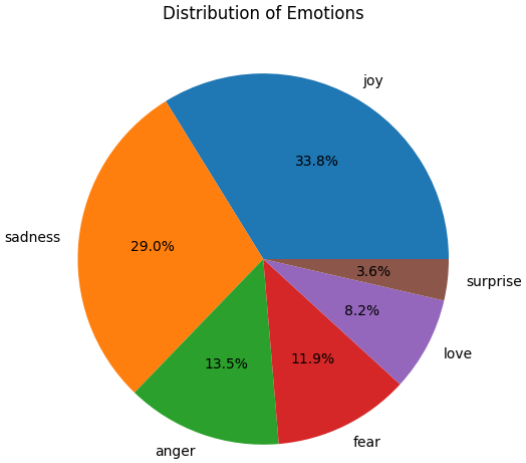
The dataset can be found on Kaggle at this link.

Data Visualization

To better understand the distribution of emotions in the dataset, several visualizations were created:

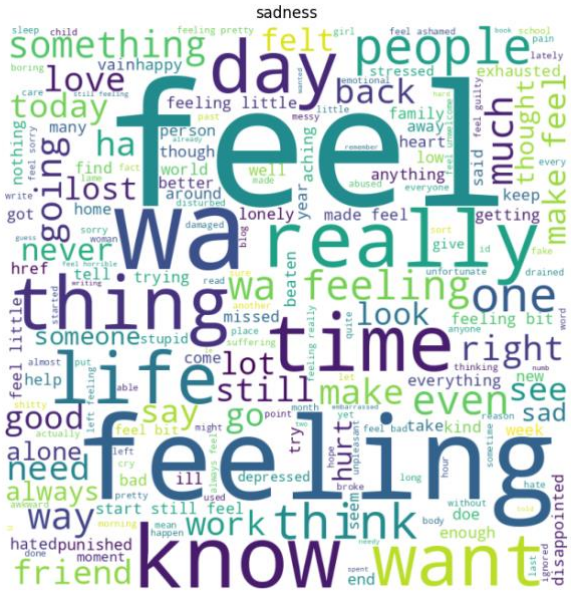
- **Bar Plot and Pie Chart**: These plots show the frequency of each emotion in the dataset. For instance, 'joy' is the most common emotion, appearing 6,761 times, while 'surprise' is the least common, appearing only 719 times. The pie chart further illustrates these proportions, showing that 'joy' makes up over 30% of the dataset, while 'sadness' is almost 30%.

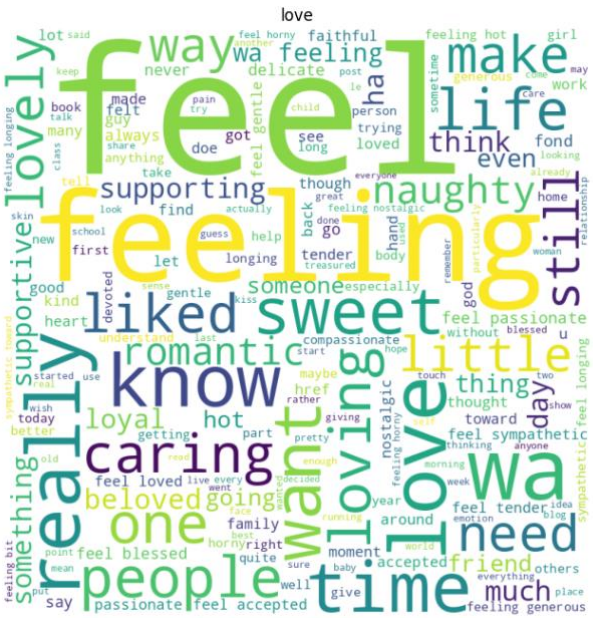


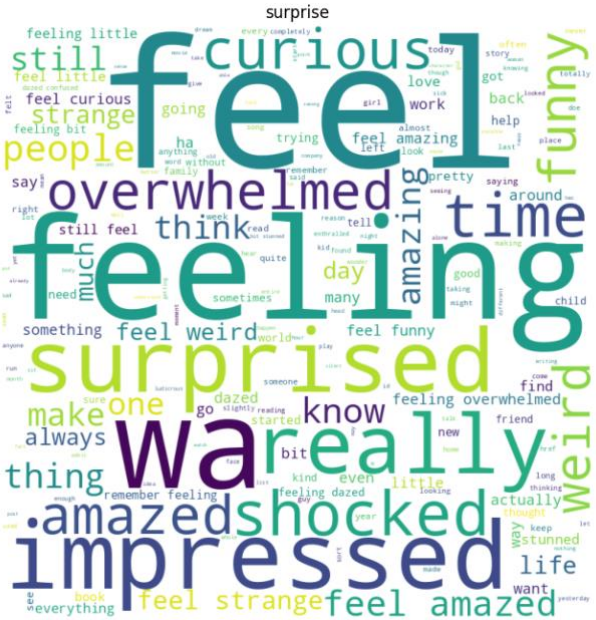


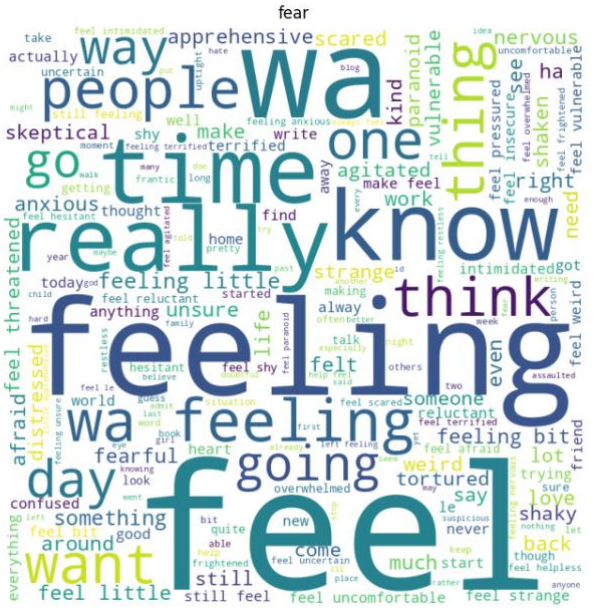
- **Word Cloud**: The word cloud shows the most frequently occurring words for each emotion. The size of each word indicates its frequency. This visualization helps to identify the key words associated with each emotion.

These visualizations provide valuable insights into the data and can guide the choice of preprocessing steps and models.









Data Preprocessing

Three different datasets were combined for this project. The text data was preprocessed using several steps:

- Removing punctuation: All punctuation was removed from the text.
- Tokenization: The text was split into individual words using the word_tokenize function from the NLTK library.

- Lemmatization: Words were lemmatized using the WordNetLemmatizer from the NLTK library. This process reduces words to their base or root form.
- Removing stop words: Commonly used words (like 'the', 'a', 'an', etc.) that do not contain important meaning were removed from the text.

The preprocessed text was then vectorized using the TF-IDF method.

Model Training and Evaluation

The preprocessed data was split into a training set and a test set. Four different machine learning models were trained on the data: Logistic Regression, Random Forest, Naive Bayes, and Decision Tree.

The performance of the models was evaluated using several metrics: precision, recall, f1_score, accuracy, and AUC. The results are as follows:

Model	Precision	Recall	F1 Score	Accuracy	AUC
Logistic Regression	87.55%	87.48%	87.11%	87.48%	98.9%
Random Forest	88.5%	88.58%	88.47%	88.58v	90.6%
Naive Bayes	41.24%	34.58%	35.85%	34.58%	61.9%
Decision Tree	86.77%	86.6%	86.66%	86.6%	90.6%

Conclusion

Based on the results, the Logistic Regression model performed the best with the highest AUC score (98.87%), indicating a high ability to distinguish between different emotions. The Random Forest model also performed well, with the highest precision (88.5%), recall (88.58%), f1_score (88.47%), and accuracy (88.58%) among all models. The Naive Bayes model performed the worst among all models, with the lowest scores in all metrics. The Decision Tree model had performance metrics similar to those of the Logistic Regression model, but its AUC (90.73%) score was lower.

In conclusion, the Logistic Regression model seems to be the most suitable for this task when using TF-IDF for feature extraction. However, the choice of model can depend on the specific requirements of the task. For example, if precision is more important than recall for your task, you might prefer the Random Forest model.

Thao Tran. Thanks for visiting.