MERCY UNIVERSITY

"CAPSTONE PROJECT"

PROJECT PROPOSAL

Name: Ravisankar Chengannagari

Advisor: Professor Tianyu Wang

Submitted to

Professor Narasimhaswamy Banavara

EMOTION CLASSIFICATION IN TWITTER DATA

ABSTRACT

I chose this project because understanding the emotions behind written text is more important in our digital world. We can see text data everywhere from customer reviews to social media posts. We can get valuable insights if we know the emotional tone of text data. This project is all about building a smart system which can automatically read a piece of text and figure out the emotion expressed in the text. The machine will be trained to recognize six basic categories of emotions: joy, love, fear, sadness, anger, and surprise. So, in this project, the main goal is to create a working machine learning model which can accurately classify emotions from new unseen text data. I will also use different machine learning algorithms to see which one gives us best result for this kind of task. It will be a very useful project for businesses which are interested in knowing their customer's feedback.

DESCRIPTION

This section provides a detailed breakdown of the project, from background research and the dataset to the methods and tools that will be used for completion.

BACKGROUND AND RELATED WORK

This project falls under the field of Natural Language Processing (NLP), particularly focusing on emotion detection. Unlike basic sentiment analysis, which only separates text into positive or negative categories, emotion detection tries to identify specific feelings like happiness, anger, or sadness.

Earlier research in this area mainly used traditional machine learning methods like Support Vector Machines (SVM) and Naïve Bayes. These approaches, often combined with techniques like TF-IDF to represent text, worked well in recognizing emotions and provided useful starting points for the field [2]. However, these methods primarily analyze word frequencies and often struggle to capture the deeper contextual meaning that comes from word order.

More recent advancements have been driven by deep learning. The introduction of models like Long Short-Term Memory (LSTM) networks, a type of Recurrent Neural Network, marked a significant leap forward. LSTMs are designed to process sequential data, enabling them to understand the context of a word based on its position in a sentence, which leads to more accurate emotion prediction. The current state-of-the-art in NLP is dominated by even more sophisticated architectures like Transformers (e.g., BERT), which have further advanced the ability of models to comprehend complex linguistic patterns by analyzing the relationships between all the words in a text simultaneously[3].

DATASET

In this project, I will use the Emotion dataset which is available on Kaggle platform.

Link: https://www.kaggle.com/datasets/nelgiriyewithana/emotions

It contains a CSV file, and the file has nearly 400,000 text entries. It has two column attributes:

1. Text: It is a string feature representing the content of the Twitter message [1].
2. Label: A classification label indicating the primary emotion, with values ranging from 0 to 5 [1].

It is a large and good dataset for training a machine learning model because it provides a wide variety of examples for each emotion which helps the model learn better.

DATA WRANGLING AND CLEANING

Basically, raw data is often messy which is available on the internet, and we cannot use it directly for training a machine learning algorithm. So, we need to clean and prepare the text data for training. I am going to perform the following steps:

1. Lowercase Conversion: I will convert all the text to lowercase so the words like 'Angry' and 'angry' treated as same word.
2. Removing Punctuation and Special Characters: I will remove the punctuations and special characters like '@', '?', '!', and '#'. These won't help in classifying emotions and can be removed.
3. Removing Stop Words: I will also remove the stop words like 'the', 'a', 'is', and 'an'. These words won't help us and does not have any meaning in it. It will help the machine learning model to focus more on important meaningful words which helps to identify emotions.
4. Lemmatization: I will change every word to its root form like 'running' or 'ran' to its root form 'run'. So, we can group them together.
5. Handling missing values: I will check if there is any missing text or label in the dataset and remove them if necessary.

FEATURE ENGINEERING

In this project, I will use two different feature engineering techniques. One for the classic machine learning algorithms and one for the deep learning algorithms.

1. Term Frequency – Inverse Domain Frequency (TF-IDF): TF-IDF technique creates a score for each word based on its frequency in a single document and its rarity across all documents. It is a very good technique for algorithms like Linear regression and Support Vector Machine (SVM).
2. Word Embeddings: Deep learning models works good with more advanced technique like word embeddings. Basically, word embeddings convert each word into a dense numeric vector. Words with similar meanings will have similar vectors. For instance, words like happy and joyful vectors are close.

ALGORITHMS AND TECHNIQUES

I am planning to train and compare a range of models to get a full picture of which algorithms perform best. This will include the classic machine learning algorithms and a modern deep learning approach.

1. Classic Machine Learning Algorithms:
    a. Multinomial Naïve Bayes: It is a fast and simple model for text classification.
    b. Logistic Regression: It is an efficient and highly interpretable classification model.
    c. Support Vector Machine (SVM): It is a very powerful model effective in high dimensional spaces like text data.
2. Deep Learning Model:
    a. Long Short-Term Memory (LSTM) Network: Long Short-Term Memory is a type of Recurrent Neural Network (RNN). It is particularly designed to understand sequences and context. When reading a sentence, an LSTM can remember words from the beginning to better understand the meaning of words that appear later. This ability to capture long range dependencies makes it an excellent choice for understanding the nuances of emotion in text.

These algorithms allow for a robust comparison between classic, statistically based machine learning methods and modern, context aware deep learning approaches.

TOOLS AND TOOLKITS

In this project, I will use Python programming language, and I will use several open-source libraries:

a. Pandas: I will use pandas library for loading the dataset from the CSV and managing the data.
b. Natural Language Toolkit (NLTK): I will use NLTK for performing the text cleaning steps like stop word removal and lemmatization.
c. Scikit-learn: I will use scikit-learn, it provides tools for TF-IDF feature engineering, implementations of all the planned algorithms (Naïve Bayes, Logistic Regression, and SVM), and tools for evaluating model performance.
d. TensorFlow with Keras: I will use TensorFlow, a leading open-source machine learning platform, with its use-friendly Keras API. This toolkit is ideal for building, training and evaluating the LSTM neural network.
e. Matplotlib and Seaborn: It will be used for creating charts and graphs to visualize the data and the results like confusion matrix.

ASSESSING RESULTS

To check whether the models are performing well or not, I will split the emotion dataset into two parts:

1. Training set: I will use 70% of the data to train the model so that it will learns the patterns properly and performs good in unseen or real-world data.
2. Validation set: After training, I will use 15% of the data to tune the model's hyperparameters. It helps in selecting the best version of each model.
3. Testing set: I will use the remaining 15% of the data to test how well the models perform on new data they haven't seen before.

I will use the following metrics to evaluate each model:

a. Accuracy: The overall percentage of predictions that are correct.
b. Precision, Recall, and F1-Score: These metrics give a more detailed look at performance, particularly if some emotions appear more often than others in the dataset. The F1-Score, which is a balance of precision and recall, is often the most important metric for this kind of task.
c. Confusion Matrix: It is a table which will show us exactly where the models are making mistakes.

UTILITY AND APPLICATION OF RESULTS

In this project, the outcome will be a trained model for detecting emotions in text and it has many practical and real-world applications. For instance:

a. Customer Experience: Businesses can analyze customer support chats or product reviews to check whether their customer is happy or frustrated in real-time. It helps them to improve their products.
b. Social Media Monitoring: Companies and public figures can track the emotional sentiment of public posts related to their brand or campaigns.
c. Mental Health: Applications could potentially analyze journal entries to help users track their emotional well-being over time.
d. Content Recommendation: News sites or steaming services could suggest articles or movies based on the emotional tone of a user's recent activity.

REFERENCES

1. Emotion Dataset from Kaggle platform:

   https://www.kaggle.com/datasets/nelgiriyewithana/emotions

2. Opinion Mining and Sentiment Analysis, Bo Pang & Lillian Lee:

   https://www.nowpublishers.com/article/Details/INR-011

3. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,

   Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova:

   https://aclanthology.org/N19-1423.pdf