Project Report

Emotion Intensity

Version: 1.0

Last Revision Date, Time: July 5, 2024, 11:00 PM

By: Raj Narayanan B

codalab username:raj_n
codalab email_id: brajnarayanan.b@gmail.com

Contents

Abstract....3

1. Introduction

    1.1) Why this Project Report?....4

    1.2) Scope....4

    1.3) Definitions....4

2. General Description

    2.1) Product perspective....5

    2.2) Problem statement....5

    2.3) Proposed solution....5

    2.4) Further improvements....5

    2.5) Technical requirements....5

    2.6) Data requirements....5

    2.7) Tools used....6

    2.8) Hardware Requirements....6

    2.9) Constraints....7

    2.10) Assumptions....7

3. Design details

    3.1) Data Preprocessing Techniques....7

    3.2) Data Loading....7

    3.2) Evaluation....8

    3.2) Modeling....8

4. Performance

    4.1) Application compatibility....8

5. Conclusion....8

Abstract:

Existing emotion datasets are mainly annotated categorically without an indication of degree of emotion. Further, the tasks are almost always framed as classification tasks (identify 1 among n emotions for this sentence). In contrast, it is often useful for applications to know the degree to which an emotion is expressed in text. This is the first task where systems have to automatically determine the intensity of emotions in tweets.

1. Introduction

1.1. Why this Project Report?

The purpose of this project report is to add the necessary detail to the project description to represent a suitable model for coding. This document can be used as a reference manual for how the modules interact at a high level.

This report will:

- Present the design aspects and define them
- Describe the performance, technical requirements
- Include design features and the architecture of the project

1.2. Scope

The scope of this project involves developing a system to automatically determine the intensity of emotions in tweets, ranging from 0 to 1. This includes collecting and annotating a representative dataset, using statistical methods like Bag of Words (BOW) and TF-IDF for text representation, implementing purely statistical approaches and Neural Networks for prediction, and evaluating the system's performance using appropriate metrics.

1.3. Definitions

- *BOW*: Bag of Words
- *TF-IDF*: Term Frequency - Inverse Document Frequency
- *Word2Vec*: Word to vector representation
- GloVe: Global Vectors for Word Representation
- OLS: Ordinary Least Squares
- RNN: Recurrent Neural Networks
- LSTM: Long Short Term Memory
- GRU: Gated Recurrent Unit
- NN: Neural Network

2. General Description

2.1. Product Perspective

From a product perspective, this project aims to create a tool that can accurately quantify the intensity of emotions expressed in tweets on a scale from 0 to 1, providing valuable insights for social media analysis, sentiment tracking, and customer feedback assessment. The tool will use statistical methods and neural networks for text representation and prediction.

2.2. Problem Statement

Given a tweet and an emotion X, determine the intensity or degree of emotion X felt by the speaker -- a real-valued score between 0 and 1. The maximum possible score 1 stands for feeling the maximum amount of emotion X (or having a mental state maximally inclined towards feeling emotion X). The minimum possible score 0 stands for feeling the least amount of emotion X (or having a mental state maximally away from feeling emotion X). The tweet along with the emotion X will be referred to as an instance. Note that the absolute scores have no inherent meaning -- they are used only as a means to convey that the instances with higher scores correspond to a greater degree of emotion X than instances with lower scores

2.3. Proposed Solution

The Emotion Intensity prediction model utilizes Statistical models and Neural Nets to ascertain whether an emotion is intense or not. By leveraging historical data and features related to tweets, the model provides an approach to reliability.

2.4. Further Improvements Possible

- Potential avenues for further improvements include the exploration of problem statement using GenAI tools and other Classical Machine Learning models.
- The design and display of results is done in the terminal itself but could have been given a user interface as well.
- The training results produced by the models could have been stored as a csv file for future inference.
- Another shortcoming was that the golden data or true labels for the Anger data was not provided - therefore it was not possible to check the metrics for that data alone.

2.5. Technical Requirements

Technical requirements include specific hardware specifications, software dependencies, and libraries needed for model development, training, and deployment. These details are crucial for ensuring the smooth execution of the project. VSCode and Python are the requirements.

2.6. Data Requirements

Ensuring robust model training, the project requires training data in a tabular format for text data. Data preprocessing has been done to ensure that the data supplied to the model is complaint.

2.7. Tools Used

Various tools and frameworks are employed, such as

- Python(v3.11.7)
- NumPy(v1.24.3)
- Pandas(v2.1.4)
- Scikit-learn(v1.2.2)
- Tensorflow(v2.13.1)
- Re(v2023.10.3)
- Scipy(v1.11.4)
- Statsmodels(v.0.14.0)
- Keras(v2.13.1)
- VSCode is used as IDE
- NLTK(v3.8.1)
- Gensim(v4.3.0)
- Glob
- Emoji(v2.12.1)
- Pickle
- Pathlib(v1.0.1)

2.8. Hardware Requirements

2.8.1. Memory (RAM):

Allocate a sufficient amount of RAM to accommodate the operational demands of the predictive model. A minimum of 16 GB or higher is recommended for handling multiple requests simultaneously.

2.8.2. Storage:

Deploy a high-capacity SSD for storage to support efficient data access and retrieval during prediction tasks. The storage should be scalable to handle growing datasets over time.

2.8.3. Network Connectivity:

Ensure a stable and high-speed internet connection, particularly if the model is hosted in a cloud environment or if there is a need for remote access.

2.8.4. Security Measures:

Implement security measures, such as firewalls and encryption, to safeguard the system and the data being processed. This is crucial, especially if the system deals with sensitive information.

2.8.5. Scalability Considerations:

If anticipating increased usage or a growing user base, design the infrastructure with scalability in mind. This may involve load balancing mechanisms and the ability to scale horizontally by adding more servers.

2.8.6. Operating System and Dependencies:

Ensure that the server environment is equipped with the necessary software dependencies and compatible operating systems for hosting the model. Linux distributions like Ubuntu Server are commonly used for deployment.

3. Design Details

3.1. Data Preprocessing Techniques

The process flow involves data cleansing, and feature extraction. In the data preprocessing techniques, the following steps are taken:

- Convertion to lowercase
- Removal of:
  - Whitespace charecters
  - All other Special Charecters
  - Stopwords
  - Emojis
- Tokenization, Stemming, Lemmatization
- Feature Extraction of stemmed and lemmatized text using:
  - Bag of Words
  - TF/IDF
  - Word2Vec
  - GloVe

The class for cleansing the data is: text_preprocess(). In this class, we will perform the basic text preprocessing and also perform Tokenization, Stemming and Lemmatization.
Both stemming and Lemmatization are performed to check the best tokenization strategy.
Once, tokenization is complete, feature extraction is performed using 4 popular methods: BOW, TF-IDF, Word2Vec, GloVe. These feature extracted data are returned together in a tuple for further processing to modelling.

Provision to return the processed dataframe alone devoid of the feature extracted data is also given so that the neural network can make use of the cleaned data alone. This is achieved from the module: text_preprocessing.py

3.2. Data Loading

There are 4 different types of data to be analysed - Anger, Joy, Fear, Sadness. We will train the statistical and the NN models separately for all these datasets. Therefore, with the number of types of datasets being 4, with the number of overall models being tested being 4 (1-statistical and 3NN), with the Stemming and Lemmatization and various feature extraction techniques being used; we will have a total of 56 model combinations for each datatype, tokenization type, feature extraction type. The module that accomplishes the task of data loading from the text file is: data_loader.py

3.3. Evaluation

The models' predictions are evaluated against the true scores or golden scores based on the pearson correlation and spearman correlation. The module that accomplishes this is: evaluate.py

3.4. Modelling

In this section, we have two categories - 1) Statistical 2) Neural Nets

In the statistical case, we are using OLS model - it is regression model and it has other counterparts like GLS, WLS, GLM - these models produced the exact same result as OLS.

- The statistical modeling is achieved by the module: statmodel.py. In this module, we have functions to fit the model, save the model, produce report from evaluation and also methods to facilitate prediction of test-data.
- The NN modeling is achieved by the module: nn_model.py. In this module, we have functions to split the data, stem and lemmatize the data, vectorize the data, build the model for training, produce reports from training, and method for prediction purpose as well.

All the models produced (OLS & NN) are saved in a folder (models) - so that they can be retrieved for future predictions. We can also version these models using model registries so that if the models are re-trained, their versions can be updateded - this way we can monitor the functioning of the models. Alongwith the models, the vectorizers are also pickled.

4. Performance

It is seen that the OLS model performs slightly better that the neural nets overall. This is because for the Neural Nets to perform good, they need more data and since the dataset is small - the models tend to overfit and we are required to regularize them - which results in underperformance.

4.1. Application Compatibility

Python acts as the central interface, ensuring seamless information transfer among different components of the project. This promotes compatibility and interoperability.

5. Conclusion

In conclusion, we can say that the OLS model can be considered as the final model for this dataset. If we had more data, we will be able to train the models for a better performance.