**MoodifyMe**

**Team Members**

* CHAMSEDDINE Lea
* EDDE Chiara
* HANNA Marielouise

**GitHub Repository**: <https://github.com/marielouisehanna/Machine_learning>

1. **Introduction**

We are working on a text classification project that focuses on detecting emotions through journal entries. The dataset, taken from the GoEmotions project, includes thousands of user-generated comments, each labeled with one or more emotions. The ultimate goal is to train a machine learning model capable of automatically identifying these emotions from raw text allowing us to suggest movies and songs accordingly to the user.

1. **Preprocessing.ipynb – Emotion Detection Data Preparation**
2. **Data Loading**

Three separate datasets (train.tsv, dev.tsv, and test.tsv) are loaded, then combined into a single one.

**2. Initial Exploration**

It’s discovered that each comment may contain multiple emotions. Also, a histogram of label frequencies shows imbalanced class distribution (some emotions are more common than others).

**3. Label Processing – One-Hot Encoding**

Each comment can have multiple emotion labels. To prepare the data for machine learning:

* Emotion IDs are mapped to their string names using a dictionary.
* A one-hot encoding approach is used

**4. Text Cleaning and Preprocessing**

To make the raw Reddit comments clean and consistent we expanded the contractions, converted emojis into words, removed URLs, hashtags, mentions and punctuation, converted the text to lower case, and applied tokenization (split text into individual words) and lemmatization (converted words to their base form with nltk), ensuring the model isn’t confused by noise or irrelevant patterns.

**5. Splitting the Dataset**

The combined dataset is split back into train and test sets using train\_test\_split(),

**6. Labels and Words Analysis:**

We created labels and performed data analysis, ensuring emotion proportions in validation and test sets matched the training set. We also analyzed mean sample lengths per emotion and identified frequent words for each emotion..

1. **From\_Scratch Model:**

**1. Model Implementation** A neural network architecture was designed for multi-label emotion classification using TensorFlow/Keras with embedding layer (128 dimensions), bidirectional LSTMs, dropout layers (20%), and sigmoid activation.

**2. Training and Evaluation** Model was compiled with binary cross-entropy loss and Adam optimizer, using early stopping, achieving 46.5% test accuracy with imbalanced performance across emotions.

**3. Analysis and Limitations** Classification reports revealed varying precision and recall across emotions (strong on joy/gratitude, weak on anger/fear) despite using techniques to prevent overfitting.

1. **Bert\_Model1:**

This notebook implements a fine-tuned BERT model to detect emotions in text using the GoEmotions dataset, which includes 28 distinct emotion labels. The goal is to perform multi-label classification and evaluate performance on both the GoEmotions taxonomy and a simplified Ekman taxonomy (7 emotions).

**Key Steps**

**1. Data Preparation**

* **Datasets**: Loaded preprocessed training, validation, and test datasets
* **Taxonomies**:
  + **GoEmotions**: 28 emotions (e.g., admiration, anger, curiosity, neutral).
  + **Ekman**: 7 emotions (anger, disgust, fear, joy, sadness, surprise, neutral).

**2. Model Configuration**

* **BERT Base Model**: Used bert-base-uncased with a sequence length of 48 (determined by the longest text in the dataset).
* **Tokenizer**: BertTokenizerFast convert text to input IDs, attention masks, and token type IDs.
* **Architecture**:

**1. Input layers** for input\_ids, attention\_mask, and token\_type\_ids.

**2. BERT Encoder**

**Purpose**: Processes input text to generate contextual embeddings (numerical representations of text that capture meaning and relationships between words).

**How It Works**: Input text is tokenized into input\_ids, attention\_mask, and token\_type\_ids. BERT’s transformer layers analyze these tokens to produce embeddings, leveraging pre-trained knowledge of language structure.

**3. Dropout Layer**

**Purpose**: Prevents overfitting.

**How It Works**: Randomly "drops" (deactivates) a fraction of neurons during training.

Forces the model to rely on diverse features rather than specific neurons.

**4. Dense Layer with Sigmoid Activation**

**Purpose**: Maps BERT’s embeddings to **multi-label predictions**.

**How It Works**:

**Dense Layer**: A fully connected layer with 28 neurons (one per emotion).

**Sigmoid Activation**: Converts raw outputs to probabilities between 0 and 1 for each emotion.

**Why Sigmoid**: It allows multiple emotions to be predicted simultaneously.

Each emotion is treated as an independent binary classification task.

**3. Training**

* **Class Weights**: Computed to address class imbalance using sklearn.utils.class\_weight.
* **Custom Loss Function**: Weighted binary cross-entropy to prioritize underrepresented classes.
* **Optimizer**: Adam with a learning rate of 5e-5.
* **Training**: 4 epochs on batches of 16 samples, with validation after each epoch.

**4. Evaluation**

* **Threshold Optimization**: Tested thresholds (0.7–0.99) to maximize macro F1-score. Best threshold: **0.92**, yielding:
  + **GoEmotions**: Macro F1 = 0.45.
  + **Ekman** (mapped from GoEmotions): Macro F1 = 0.58.
* **Handling Empty Predictions**:
  + Assigned "neutral" to samples with no predicted emotions.
  + Improved precision/recall balance but had minimal impact on F1.

**6. Predictions and results**

* **GoEmotions**: Achieved moderate performance with high recall (0.68) but lower precision (0.34), indicating a tendency to over-predict emotions.
* **Ekman**: Better performance due to label consolidation, with "joy" and "fear" showing strong F1-scores (0.80 and 0.60, respectively).

**7. Challenges & Insights**

**- Class Imbalance**: Rare emotions had low precision due to sparse training examples.

**- Threshold Sensitivity**: High thresholds reduced false positives but increased empty predictions.

**- Neutral Label Ambiguity**: Some "neutral" samples contained detectable emotions (label noise).

1. **Bert\_Model2:**

This notebook is similar to the previous one, but with one important change: we removed the 'Neutral' samples from the dataset before training the model.

* By removing the neutral comments, the model was able to focus better on recognizing real emotions like happiness, sadness, or anger. This helped improve the model’s ability to distinguish between different emotions.
* Interestingly, even though we didn’t directly teach the model to recognize neutral comments, it was still able to detect them. Since the model was only trained on emotional comments, it naturally learned to identify neutral comments as something different, without needing to be explicitly told what “Neutral” looks like.

1. **User Interface and Databases:**

Since development is not the main focus of the project, we used XAMPP, a local server environment that lets us run web applications on our computer with no need for internet or an actual server online. We created a friendly interface to better visualize our project and to be able to test and use our models and codes in a fun, clear and easy way.

Xampp includes MySQL for the database, and we used it to develop our databases:

* Movies database containing each movie’s title, director, genre and “mood” that corresponds to one of the 27 emotions that can be predicted
* Songs database is divided similarly into title, artist, genre and mood.