Sentimental analysis - Challenge 3 AML

Alex Argese
Student
EURECOM
Biot, France
alex.argese@eurecom.fr

Cristian Degni
Student
EURECOM
Biot, France
cristian.degni@eurecom.fr

Miriam Lamari
Student
EURECOM
Biot, France
miriam.lamari@eurecom.fr

Enrico Sbuttoni
Student
EURECOM
Biot, France
enrico.sbuttoni@eurecom.fr

***Abstract*—**

- **Sentence-level Sentiment Analysis evaluate sentiment from a single sentence.**

***Index Terms*—Machine Learning, Sentimental Analysis, Natural Language Process, ,**

I. Introduction

Sentiment analysis is the process of determining the opinion, judgment, or emotion behind natural language. It can be a can be a very powerful technique since it is widely applied to voice of the customer materials such as reviews and survey responses, online and social media. The most advanced sentiment analysis can identify precise emotions like anger, sarcasm, confidence or frustration. In this challenge, we focus on sentence-level sentiment analysis, which evaluates sentiment from a single sentence. The primary goal is to classify sentences into one of three categories: **positive**, **negative**, or **neutral**. The dataset used for this task consists of **tweets from Figure Eight's Data** for Everyone platform.

II. Dataset

The dataset is provided in CSV format, with each row representing a tweet and its associated metadata. The training set contains **24732 samples**, while the test set contains **2748 samples**. The dataset is available on the **Figure Eight** platform, which provides a wide range of datasets for various machine learning tasks. Inside training set, four columns are present:

- **textID**: unique identifier for each tweet;
- **text**: the tweet text;
- **selected_text**: the text selected by the annotator as the most relevant part of the tweet for sentiment analysis;
- **sentiment**: the sentiment label assigned to the tweet, which can be one of three classes: **positive**, **negative**, or **neutral**.

The **class distribution** within the training set is **unbalanced**:

- **positive**: 7711 samples
- **negative**: 7003 samples
- **neutral**: 10018 samples

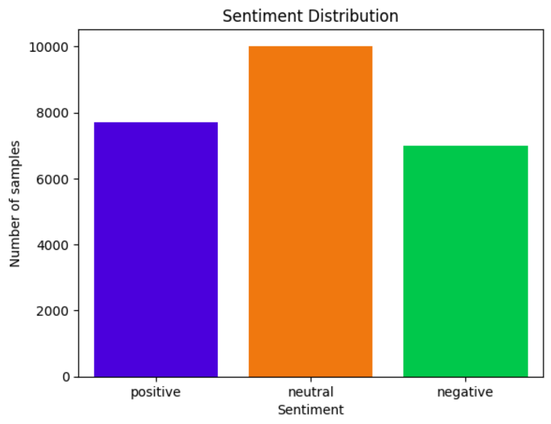


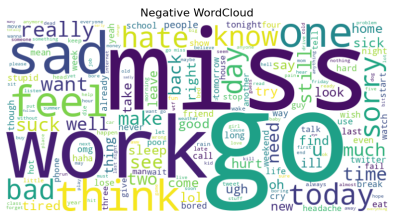
Fig. 1: Train samples class distribution.

III. Preprocessing

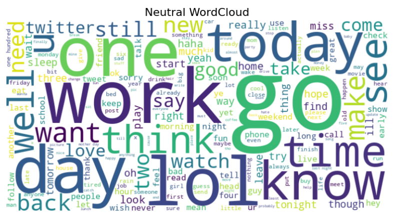
In order to prepare the dataset for training, we performed several preprocessing steps on the text data:

- **removing URLs, mentions and hashtags**: URLs, mentions, and hashtags were removed from the text to focus on the actual content of the tweet;
- **removing punctuation and special characters**: punctuation marks and special characters were removed to simplify the text and reduce noise;
- **conversion of numbers into words**: numbers were converted into their word equivalents to maintain consistency in the text;
- **lowercase**: all text was converted to lowercase to ensure uniformity and avoid case sensitivity issues;
- **removing stop words**: common words that do not contribute to the sentiment analysis were removed to reduce noise and improve model performance. Words of negation such as "not" and "no" were kept, as they can significantly impact the sentiment of a sentence;
- **tokenization**: the text was tokenized into individual words to facilitate further processing and analysis.

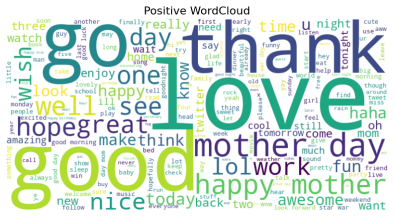
It follows the WordCloud visualization of the most frequent words in the training set, depending on the class they belongs to, which highlights the most common terms used in the tweets. This visualization can help identify key themes and topics present in the dataset.



,



,



,

Fig. 2: Word clouds for the negative, neutral, and positive classes.

IV. Models Evaluated

A. GRU-Based Sentiment Classifier

A **Gated Recurrent Unit (GRU)**-based model was developed using **TensorFlow/Keras** to classify **tweet sentiment** from **preprocessed text data**. Each input is first transformed into padded token sequences (max_len = 50) and passed through an **Embedding layer** (output_dim=128). The embedded sequences are then processed by a **GRU layer** with **64 units**, followed by a **Dropout layer** (rate=0.5) to mitigate overfitting. A final **Dense layer** with **softmax activation** predicts one of the **three sentiment classes**.

The training process uses **sparse categorical crossentropy** as the loss function and the **Adam optimizer** with a **batch size of 32** for **10 epochs**. Model evaluation on the test set includes computing the **macro F1 score** and generating a **detailed classification report**. Labels are encoded using **LabelEncoder** to match the expected format for training.

TABLE I: Hyperparameter setup for GRU-based Sentiment Classifier

Hyperparameter	Ranges / Values	Used
embedding_dim	128	128
gru_units	32, 64, 128	64
dropout	0.3–0.5	0.5
optimizer	Adam	Adam
epochs	10–30	10
batch_size	32, 64	32

B. RNN-Based Sentiment Classifier

A **Recurrent Neural Network (RNN)**, specifically a **LSTM-based classifier**, was implemented using **TensorFlow/Keras** to perform **sentiment classification** on **preprocessed tweet data**. The architecture begins with a **tokenized and padded input sequence** (max length = **50**), embedded into a **dense vector space** via an **Embedding layer** (output_dim=128). This is followed by a **single LSTM layer** (units=64) and a **Dropout layer** (rate=0.5) to reduce overfitting. The final **Dense output layer** uses a **softmax activation** for multi-class classification across **three sentiment labels**.

The model is trained with **sparse categorical crossentropy** loss and optimized using the **Adam optimizer** over **10 epochs**. Performance is evaluated using **macro-averaged F1 score** and a detailed **classification report**, with results obtained on a held-out **test set**. Input labels are encoded numerically using **sklearn's LabelEncoder** for compatibility with the loss function.

TABLE II: Hyperparameter setup for LSTM-based Sentiment Classifier

Hyperparameter	Ranges / Values	Used
embedding_dim	128	128
lstm_units	32, 64, 128	64
dropout	0.3–0.5	0.5
optimizer	Adam	Adam
epochs	10–30	10
batch_size	32, 64	32

C. RNN with Self-Attention Sentiment Classifier

A **Recurrent Neural Network** enhanced with a **Self-Attention mechanism** was constructed using **TensorFlow/Keras** to classify **tweet sentiment**. Each input sentence is tokenized and padded (max_len = 50), then embedded via an **Embedding layer** (output_dim=128). This is followed by a

GRU layer (units=64) whose output is fed into a **custom Self-Attention layer**. The attention mechanism computes **context-aware weighted representations** of the input sequence to improve the model's focus on relevant tokens. The result is aggregated and passed to a **Dense softmax layer** for final classification into **three sentiment classes**.

The model is trained using the **Adam optimizer** and **sparse categorical crossentropy** loss function for **10 epochs**, with **batch size 32**. Labels are numerically encoded using **LabelEncoder**. The classifier's performance is assessed through a **macro-averaged F1 score** and a **classification report** on a held-out test set.

TABLE III: Hyperparameter setup for RNN with Self-Attention Sentiment Classifier

Hyperparameter	Ranges / Values	Used
embedding_dim	128	128
gru_units	32, 64, 128	64
attention	Custom Layer	✓
optimizer	Adam	Adam
epochs	10–30	10
batch_size	32, 64	32

D. TF-IDF + Traditional Classifiers for Sentiment Analysis

A **classic machine learning pipeline** was employed to classify **tweet sentiment** using a **TF-IDF vectorization** of preprocessed text. Input text is tokenized, cleaned, and then transformed into a **TF-IDF matrix** (max_features=5000) using **scikit-learn's TfidfVectorizer**. The resulting sparse matrix represents word importance across documents and serves as input to various **supervised classifiers**.

Multiple models were evaluated, including **Logistic Regression**, **Linear SVM (SVC)**, **Multinomial Naive Bayes**, and **Random Forests**. Each model is trained on **70% of the data**, validated on **15%**, and tested on the remaining **15%**. The primary metric for evaluation is the **macro-averaged F1 score**, complemented by **classification reports** and **confusion matrices**.

TABLE IV: Hyperparameter setup for TF-IDF + Traditional Classifiers

Component	Details	Used
vectorizer	TfidfVectorizer	✓
max_features	1000–10000	5000
classifiers	LogReg, SVM, RF, NB	RF

E. GloVe + LSTM Sentiment Classifier

A **GloVe-augmented LSTM model** was developed for **tweet sentiment classification**, combining **pretrained word embeddings** with a standard **LSTM architecture**. Input texts are first tokenized and padded to a **fixed length of 100**, then mapped to **100-dimensional GloVe vectors** (glove.6B.100d) to incorporate **semantic word relationships**. These embeddings are fed into a **single LSTM layer**, followed by a **dropout** and **dense softmax output** for multi-class classification.

The model was trained using **sparse categorical crossentropy** and the **Adam optimizer**, with training/validation/ test splits of **70%/15%/15%**. The use of **pretrained embeddings** enabled better generalization, particularly for rare or out-of-vocabulary tokens. Final evaluation was performed using **macro F1 score** and a **detailed classification report**, revealing robust performance across sentiment classes.

TABLE V: Hyperparameter setup for GloVe + LSTM Sentiment Classifier

Hyperparameter	Ranges / Values	Used
embedding_type	GloVe 6B 100d	✓
embedding_dim	50, 100, 200	100
lstm_units	32, 64, 128	64
dropout	0.3–0.5	0.5
optimizer	Adam	Adam
batch_size	32, 64	64
epochs	5-20	5

F. RoBERTa-Based Sentiment Classifier

A **Transformer-based model**, specifically **RoBERTa fine-tuned for sentiment analysis**, was implemented using the **Hugging Face Transformers** library. The model used is "cardiffnlp/twitter-roberta-base-sentiment", pre-trained on social media text. Input sentences are first **tokenized** using the corresponding **AutoTokenizer**, padded and truncated appropriately, and then passed to the **RoBERTa model** which outputs contextualized representations. A classification head maps these to **three sentiment classes**.

The model is trained using the **Trainer API** with **weighted F1 score**, **accuracy**, **precision**, and **recall** as evaluation metrics. Data is managed using the **Hugging Face Datasets** library and split into **70% training**, **15% validation**, and **15% test** sets. Labels are encoded via **LabelEncoder** to match the model's expected format.

TABLE VI: Hyperparameter setup for RoBERTa Sentiment Classifier

Hyperparameter	Ranges / Values	Used
batch_size	8, 16, 32	default
epochs	3–5	default
max_length	auto via tokenizer	✓

V. Metric Justification

To evaluate model performance in ranking anomalies, we primarily used the **ROC AUC**, which measures the model's ability to distinguish between **normal** and **anomalous** samples, independently of a decision threshold. This is particularly important in **unsupervised anomaly detection**, where decision boundaries are not known a priori.

While additional classification metrics such as **F1-score**, **precision**, and **recall** were reported after threshold selection, they were not used for **model selection** or hyperparameter tuning.

VI. Results Summary

A. Detailed Results – GRU Classifier

The **GRU-based model** obtained a lower **macro F1 score of 0.6304**, showing decreased precision and recall, particularly on the **neutral** class (F1 = 0.56). Despite the architectural efficiency of GRUs, their performance on this dataset did not surpass LSTM-based models, highlighting the sensitivity of sentiment classification to subtle sequential dependencies.

B. Detailed Results – RNN Classifier

The **Recurrent Neural Network (RNN)** with a single **LSTM layer** achieved a **macro F1 score of 0.6755** on the test set. While the model performed reasonably across all sentiment classes, its recall for **neutral** sentiment (0.60) lagged slightly, indicating difficulty distinguishing non-polar content. Nevertheless, the model demonstrated stable performance, especially on **positive** samples, and serves as a robust baseline.

C. Detailed Results – RNN + Self-Attention

Integrating a **Self-Attention layer** on top of GRUs resulted in a **macro F1 score of 0.6924**, the highest among the custom RNN architectures. Notably, the **positive class** achieved an F1 of 0.76, with the **neutral** class also improving compared to previous models. The attention mechanism clearly enhanced the model's ability to focus on relevant contextual tokens, leading to more balanced predictions.

D. Detailed Results – RoBERTa Classifier

The **pretrained RoBERTa model** (cardiffnlp/twitter-roberta-base-sentiment) significantly outperformed all other approaches, achieving a **macro F1 score of 0.8110** and an **overall accuracy of 81%**. Precision, recall, and F1 were highly consistent across all classes, with especially strong performance on **positive** sentiment (F1 = 0.85). This confirms the power of **transfer learning** and contextual embeddings for handling informal, user-generated content like tweets.

E. Detailed Results – GloVe + LSTM

The **GloVe + LSTM** model combined **pretrained GloVe embeddings** (glove.6B.100d) with a classic **LSTM architecture** to classify tweet sentiment. This approach achieved a **macro F1 score of 0.7134** on the test set. The model performed consistently across all classes, with particularly balanced results for **neutral** (F1 = 0.69) and **positive** (F1 = 0.76) sentiments. Its ability to integrate **semantic information** from pretrained vectors allowed it to well generalize.

F. Detailed Results – TF-IDF + Random Forest

Using a **TF-IDF representation** (max_features=5000) combined with a **Random Forest classifier**, this classical machine learning setup achieved a **macro F1 score of 0.7125**. The model performed particularly well on the **positive** (F1 = 0.76) and **neutral** (F1 = 0.70) classes, demonstrating that, with well-engineered features, traditional ensemble methods can still be competitive. Although less flexible than deep architectures, this approach offers robustness, simplicity, and high interpretability.

G. Metric Selection Rationale

All models were evaluated using the **macro-averaged F1 score** to fairly represent performance across imbalanced classes. While traditional classifiers benefited from interpretability and simplicity, **deep learning** models—especially those with attention or pretraining—demonstrated superior generalization and robustness. **RoBERTa**, in particular, shows the effectiveness of leveraging large-scale pretrained language representations for fine-grained sentiment understanding.

TABLE VII: Performance Comparison of Models on the Test Set

Model	F1 Macro
RNN (LSTM)	0.6755
GRU	0.6304
RNN + Self-Attention	0.6924
TF-IDF + Random Forest	0.7125
GloVe + LSTM	0.7134
RoBERTa (pretrained)	0.8110

VII. Model Selected

Although both the **TF-IDF + Random Forest**, **RNN + Self-Attention** and **GloVe + LSTM** models performed reasonably well—achieving **macro F1 scores** of **0.7125** and **0.6924** respectively—the **RoBERTa-based classifier** clearly outperformed all alternatives, reaching a **macro F1 score of 0.8110** and an **overall accuracy of 81%** on the test set.

Given the **supervised nature** of the task and the importance of capturing **contextual sentiment nuances** across informal text, **macro F1 score** was prioritized as the primary evaluation metric to ensure balanced performance across all classes.

Therefore, we selected the **RoBERTa (pretrained)** model as the **final architecture**, due to its superior **generalization ability**, **contextual understanding**, and consistent performance across all sentiment categories. This model leverages **state-of-the-art transformer representations** and benefits from **pretraining on large-scale social media data**, requiring minimal task-specific tuning while delivering **high-quality sentiment predictions**.

VIII. Inference and Future Work

A. *Inference on Unlabeled Evaluation Set*

For the final submission, we deployed our best-performing model—**RoBERTa fine-tuned for Twitter sentiment analysis**—on the unlabeled test set. Each input tweet was tokenized, encoded, and processed by the model to produce a sentiment prediction (**positive**, **neutral**, or **negative**).

Due to the model's strong performance on the validation and test sets (macro F1 = 0.8110), we expect it to generalize well, especially thanks to its pretraining on large-scale Twitter data.

B. *Possible Improvements and Creative Extensions*

While the RoBERTa-based classifier proved most effective, several potential enhancements and experimental directions can be explored to further improve performance or generalization:

- **Mixture of Experts (MoE)**: Combine predictions from different classifiers (e.g., RoBERTa, GRU+Attention, TF-IDF+SVM) via a learned gating network or rule-based selector that chooses the most appropriate model per tweet based on features like length, sentiment polarity, or linguistic complexity.
- **Prompt-based Inference with LLMs**: Use instruction-tuned large language models (e.g., Flan-T5, Mixtral, GPT-4) with prompts such as: > "The sentiment of the following tweet is [MASK]: 'Why is everything closed today?'" This allows zero- or few-shot classification and could adapt better to ambiguous or nuanced expressions.
- **Contrastive Learning for Sentiment Embeddings**: Fine-tune the sentence embeddings using a contrastive objective on pairs of tweets with similar/dissimilar sentiments to enhance representation learning for downstream classification.
- **Data Augmentation with Back-Translation and Synonym Replacement**: Enrich training data via automatic paraphrasing using back-translation (EN $\longrightarrow$ FR $\longrightarrow$ EN) and lexical substitution, which increases diversity and robustness.
- **Temporal Sentiment Tracking**: If tweet metadata is available, model sentiment trends over time (e.g., for recurring users or events) with temporal models like Transformer-based sequence encoders or hierarchical LSTMs.

IX. On Binary Simplification of the Classification Task

During the early phases of model development, we considered simplifying the problem from a **three-class sentiment classification** to a **binary task**, distinguishing only between **polar** (positive or negative) and **neutral** sentiment. The rationale was to potentially reduce class confusion and improve precision on polar classes, which are often the target in sentiment-aware applications (e.g., customer feedback analysis).

However, after analyzing the **confusion matrix** from multiple models—including the GRU, LSTM, and RoBERTa classifiers—we observed the following:

- The classifier did **not show any dominant bias** toward a particular class; predictions were distributed in a relatively balanced manner across the three sentiment categories.
- The **neutral class**, in particular, exhibited **almost symmetrical counts of false positives and false negatives**, indicating that the classifier's performance on neutral samples was neither overly conservative nor excessively lenient.
- Misclassifications tended to occur at the **semantic boundaries** between classes (e.g., between slightly negative and neutral), rather than being concentrated in one direction.

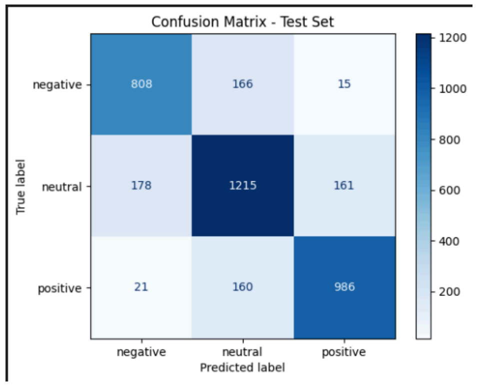


Fig. 3: Confusion matrix of RoBERTa classifier on the test set. Class predictions are balanced, with neutral showing no dominant error pattern.

Based on these findings, we decided to retain the full **multiclass setup** (positive / neutral / negative), as the model was already able to handle the three-way distinction without introducing bias or instability. Additionally, maintaining the full label set aligns better with the downstream use case of nuanced sentiment understanding in social media analytics.

X. Conclusion and Next Steps

In this challenge, we compared three unsupervised approaches for **anomalous sound detection** on the **slider machine** using audio recordings from the MIMII dataset. Our analysis revealed that while **fully-connected VAEs** were limited by their inability to capture spectro-temporal patterns, **Convolutional VAEs** significantly improved anomaly detection by leveraging local structure in Mel spectrograms.

However, the best results were obtained by the **PANNs-based model** with **Mahalanobis distance**, which achieved a **ROC AUC of 0.9311** without requiring retraining. This underscores the value of **pretrained semantic audio representations** for generalization in real-world noisy environments.

As future work, we propose exploring **semi-supervised training** using pseudo-labeling strategies, experimenting with **attention-based models** applying **temporal models** (e.g., LSTMs or Transformers) to capture longer-term dependencies in machine sounds, and evaluating the use of **domain adaptation techniques** to improve robustness across different machine types or noise conditions.

XI. References

This report is inspired by the DCASE challenge and its application to real-world industrial environments, as described in:

DCASE Challenge Task 2 (2020), **Unsupervised Anomalous Sound Detection for Machine Condition Monitoring**. The MIMII Dataset: Koizumi et al., **MIMII Dataset: Sound Dataset for Malfunctioning Industrial Machine Investigation**, 2019. The ToyADMOS Dataset: Purohit et al., **ToyADMOS: A Dataset of Miniature-Machine Operating Sounds for Anomalous Sound Detection**, 2019.