IDIAP RESEARCH REPORT

SENTIMENT ANALYSIS USING PRETRAINED LLMS

Alexandre Huou [a] Petr Motlicek
Esaú VILLATORO-TELLO [b]

Idiap-RR-05-2024

JULY 2024

[a] EPFL
[b] Idiap

Centre du Parc, Centre du Parc, Rue Marconi 19, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

Report
Bachelor Semester Project

Sentiment Analysis using pretrained LLMs

by Alexandre Huou (SCIPER 342227)
Communication Systems, BA6
8 ECTS

Supervised by
Prof. Petr Motlicek
Dr. Esau Villatoro

Contents

1 Introduction

The ubiquity of digital communication, especially on social media platforms, provides a rich source of data that reflects a wide range of human emotions. Analyzing this data to detect underlying emotional states involves challenges related to the subtleties and complexities of language, context, and cultural expressions. Furthermore, as online interactions increase, the need for automated systems to effectively manage and respond to such emotional data becomes more critical.

This project aims to evaluate general pretrained LLM's capacity in the field of Sentiment Analysis. In order to do so, 2 datasets were used:

- DS1: Text Dataset for Text Emotion Detection [1]:
- DS2: Tweets Emotion Detection [2]:

This project's code is contained in a GitHub repository [3]. The models used aren't trained or fine tuned with respect to classification, but generation instead, which will make getting good accuracy a challenging task. The prompts folder gives examples of the prompts fed in order to achieve this.

The challenge lies in balancing complexity and detail for these prompts, as the model is very quick to hallucinate and give unexpected labels, despite it needing some sort of lengthy explanation to a degree in order to best answer within the demanded classes.

By achieving these objectives, this study intends to contribute to the broader field of affective computing by providing insights and methodologies that can be employed to enhance the emotional intelligence of automated systems. This not only aids in better understanding human communications on digital platforms but also supports various applications that rely on sentiment analysis to tailor content, responses, or services according to user emotions.

2 Related Work

2.1 IDIAP's Audio Sentiment Analysis Model

This project is part of a larger IDIAP initiative aimed at detecting sentiment and emotion from audio. The broader project seeks to develop models that can interpret emotions from both audio and text sources.

The role of this specific module is to analyze text. It processes transcripts likely generated from audio files by speech-to-text technology. This allows the system to evaluate emotional content in written form, which complements the audio analysis.

This aligns with ongoing research and expands the practical uses of emotion detection technology.

3 Dataset

3.1 Dataset Preprocessing

DS1 is described as a collection of SMS and Messages, DS2 as one of Tweets. The former is given with labels in one-hot encoding:

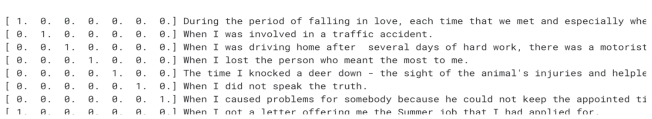
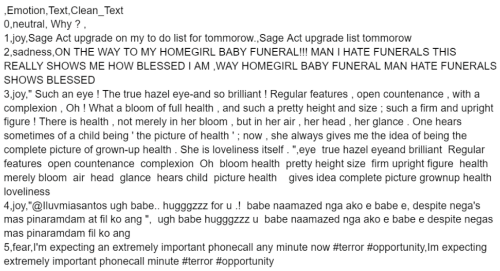


Figure 1: Extract from DS1

Thus the first task was to find out the mapping (for some reason this wasn't explicit in the kaggle, and going through the creator's personal projects was needed). Then, since the second dataset too was about emotions, a mapping transforming a selected set of emotions into sentiment was required.



```
,Emotion,Text,Clean_Text
0,neutral, Why ? ,
1,joy,Sage Act upgrade on my to do list for tommorow.,Sage Act upgrade list tommorow
2,sadness,ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN I HATE FUNERALS THIS
REALLY SHOWS ME HOW BLESSED I AM ,WAY HOMEGIRL BABY FUNERAL MAN HATE FUNERALS
SHOWS BLESSED
3,joy," Such an eye ! The true hazel eye-and so brilliant ! Regular features , open countenance , with a
complexion , Oh ! What a bloom of full health , and such a pretty height and size ; such a firm and upright
figure ! There is health , not merely in her bloom , but in her air , her head , her glance . One hears
sometimes of a child being ' the picture of health ' ; now , she always gives me the idea of being the
complete picture of grown-up health . She is loveliness itself . ",eye  true hazel eyeand brilliant  Regular
features  open countenance  complexion  Oh  bloom health  pretty height size  firm upright figure  health
merely bloom  air  head  glance  hears child  picture health  gives idea complete picture grownup health
loveliness
4,joy,"@Iluvmiasantos ugh babe.. hugggzzz for u .!  babe naamazed nga ako e babe e, despite nega's
mas pinaramdam at fil ko ang ",  ugh babe hugggzzz u  babe naamazed nga ako e babe e despite negas
mas pinaramdam fil ko ang
5,fear,I'm expecting an extremely important phonecall any minute now #terror #opportunity,Im expecting
extremely important phonecall minute #terror #opportunity
```

Figure 2: Extract from DS1

The following choice was arbitrarily made:

- joy - Positive
- disgust, sadness, fear, anger, shame, guilt - Negative
- neutral - Neutral
- Surprise - X

The decision to fully remove the "Surprise" instances from the dataset, stems from the lack of "common sense" way to map them back to either positive or negative as surprises can be both. Furthermore, given it represented a very small fraction of the overall set it wouldn't affect too negatively our results.

Once this basic comprehension-based preprocessing made, the python module data_player.py was made, in order to get a rough understanding of the datasets. This script gives basic statistics about most and least frequent apparitions of certain keywords. In order for it to have any relevance whatsoever, another kind of preprocessing was applied to the datasets directly, passing them through the ntlk module in order to remove useless words (typical stopwords include: "the", "he", "on", etc...). Here is a table that sums up the obtained information:

Common words per emotion	**anger**: [friend: 125, angry: 114, one: 78, told: 73, time: 70] **disgust:** [disgusted: 82, saw: 79, people: 79, one: 79, man: 71] **fear:** [night: 129, would: 111, home: 101, alone: 95, car: 95] **guilt:** [felt: 160, guilty: 135, friend: 101, mother: 89, one: 70] **joy:** [friend: 85, first: 79, happy: 70, time: 69, got: 69] **sadness:** [died: 126, friend: 111, sad: 111, felt: 97, time: 70] **shame:** [felt: 140, ashamed: 112, friend: 89, one: 74, time: 63]	**anger**: [angry: 202, anger: 173, time: 161, people: 150] **disgust:** [like: 85, people: 53, amp: 48, rt: 35, disgruntled: 28] **fear:** [love: 439, afraid: 425, fear: 390, tomorrow: 389, today: 386] **joy:** [amp: 640, day: 559, christmas: 468, time: 456, like: 427] **neutral:** [nan: 443, yes: 110, ok: 51] **sadness:** [like: 278, sad: 255, time: 225, cant: 177, felt: 164] **shame:** [embarrassed: 43, ashamed: 25, embarrassment: 20, embarrassed: 13,
	Simple Emotions Dataset	Twitter Emotions
Number Instances	7480 **Positive:** 1084 **Negative:** 6396	30730 **Positive:** 11045 **Neutral**: 2254 **Negative**: 17431
Average Length Instance	114.4	51.16
Vocabulary size	14486	38887
Number of tokens	166255	236504
Length Standard Emotion	74.7	29.16
Length Variance Emotion	5581.56	850.6

Figure 3: Table with statistics

4 Method Implemented

4.1 GPU Acceleration

Now that the datasets are preprocessed, the actual LLM work can start. For this project we are using the llama-cpp-python package on Windows. This made for the first (and one of the longest) bugs. The bug was really hard to diagnose, as the sole indicator was : Cublas = 0. This means simply that GPU Acceleration is off, without any traceback or error message. This git issue [4], solved the problem along with a fresh reinstall of our Nvidia GPU drivers and associated software (we also mention it in our repository).

Once GPU Acceleration was working, we started simply by doing 0-shot-prompting, that is 1 prompt per element of our dataset, of the form:

DS2:
You are an experienced Twitter analyst trained to evaluate tweet extracts.
"Return "Positive" or "Negative" or "Neutral" based on the underlying emotion"
For example: [Dataset instance]

Figure 4: 0-shot prompt for DS2

4.2 Model Misclassifications

From this point onwards, the Llama model started behaving oddly, returning the prompt given to it in form of its output (instead of one of "Positive", "Negative", "neutral"). Many hours were spent, trying to change prompts, upgrade versions, follow prompt engineering courses (namely Llama's one [5]) to no avail. On the other hand, Mistral's model worked like a charm, giving 90% of answers in the possible labels. The only (but persistent) problem is that of the model overusing neutral as a prediction, and even when evaluating it on the first dataset (that doesn't have neutral in its classes).

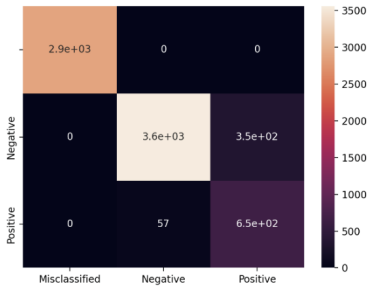


Figure 5: Confusion Matrix with all extra labels in "Misclassified"

	precision	recall	f1-score	support
Misclassified	1.00	1.00	1.00	82
Negative	0.83	0.52	0.64	17384
Positive	0.74	0.32	0.45	11011
neutral	0.13	0.88	0.23	2253
accuracy			0.47	30730
macro avg	0.67	0.68	0.58	30730
weighted avg	0.74	0.47	0.54	30730

Figure 6: Classification Report

In order to solve this problem, 2 solutions were implemented: we pass the model's output through the NLTK module once more, to convert all variations of a label back to its root, then we force any result that's still different by mapping it to the user chosen majMap.

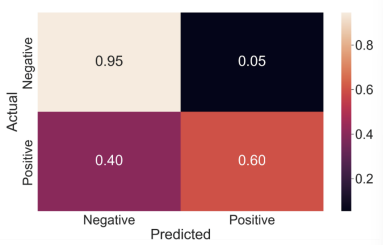


Figure 7: Confusion Matrix with all extra labels mapped to Negative for DS1

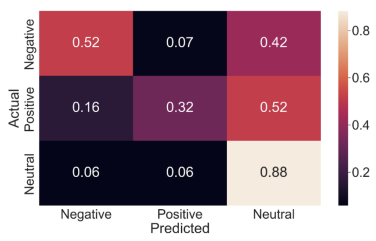


Figure 8: for DS2

4.3 Ideas for improving the accuracy

Before using few-shot learning on the model to get the accuracy up, a few ideas were considered:

- Trying completely different prompts (each run of the model takes about 3-6 hours depending on the dataset that's tested).
- Prompting to get probabilities and then doing some post processing, thus avoiding the model hallucinating However to consistently get decent results, multiple examples had to be given thus effectively doing few-shot prompting
- Fully fine tune a model to this specific task, but we simply didn't have the hardware resources for such a task.

Overall, none of these solutions seemed really promising and so the decision was made to start using 3-shot prompting instead.

4.4 3-shot prompting

The n-shot prompting technique consists in giving randomly selected examples of n desired input-output tuples. LLMs are few shot learners and it has been largely reported that by giving examples, the accuracy improves significantly. However, at first I misunderstood the task, and instead manually picked 3 examples, made a static prompt with these and fed it to the LLM. Needless to say, the accuracy didn't bulge.

At first, the prompt systematically had at least one randomly chosen example per label. However, due to the dataset's labels not following a uniform distribution, this probably biased the model and made it predict "neutral" disproportionately with respect to its order.

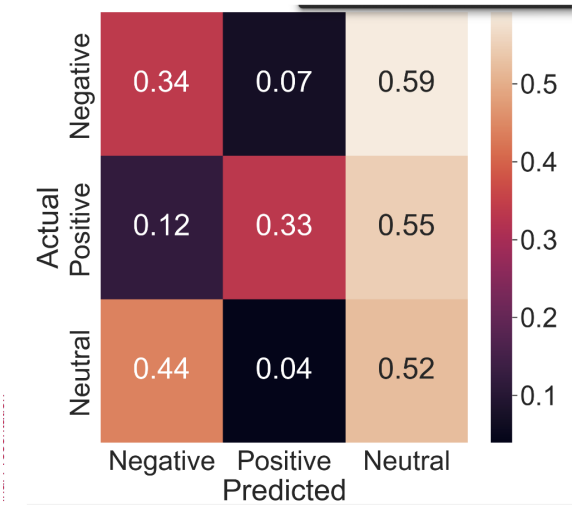


Figure 9: Confusion Matrix for DS2 with classified 3-shot

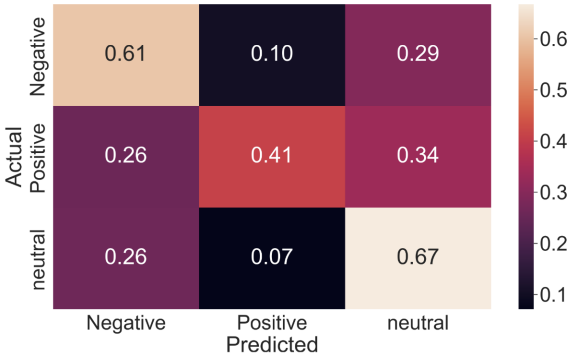


Figure 10: Confusion Matrix for DS2 with unclassified 3-shot

As we can see, when classifying the shots the model heavily predicted neutral (3rd column on both images) as it was artificially biased due to the sheer number of examples it was fed over the full run.

The codebase works flawlessly at this point, there remains only to find the best accuracy-wielding prompts, and to run baselines to compare whichever results might be obtained to what is already being done.

5 Baselines

5.1 Performance with ML Classifiers

Several traditional machine learning classifiers were used on our own datasets, including Logistic Regression, Support Vector Machine (SVM), Decision Tree, and Random Forest, to analyze their effectiveness in regards to emotion detection in text. These classical approaches were fine-tuned to handle the nuances of textual data, providing a baseline for performance metrics. These were expected to be the "lower bound", deciding whether our not fine-tuned LLM approach was worth considering.

This comparison was crucial to highlight the advancements in deep learning for natural language processing tasks and to evaluate the potential improvements offered by state-of-the-art transformer-based models over traditional methods.

5.2 Performance with Roberta

We employ the RoBERTa model, anticipating it to surpass the performance of our baseline models due to its specialized fine-tuning for emotion detection tasks. RoBERTa is an evolution of BERT (Bidirectional Encoder Representations from Transformers), which was originally developed by Google. BERT revolutionized the field of natural language processing by enabling models to understand context from both left and right sides of a word within a text, significantly enhancing the model's ability to interpret complex language constructs.

RoBERTa extends BERT's methodology by training more extensively on a larger corpus and with more refined tuning of hyperparameters. This extensive training allows RoBERTa to achieve even greater performance benchmarks across various NLP tasks, including those closely related to emotion detection in text.

Given that RoBERTa has been fine-tuned specifically for the kind of semantic analysis required in detecting emotional nuances in text, we anticipate it to demonstrate superior accuracy and effectiveness compared to our initial models, and thus be our "higher bound".

5.3 Results

The comparative performance between baseline classifiers and our RoBERTa-based model underlines key insights into the adaptability and robustness of these approaches to emotion detection in text. Our model shows a consistent performance that lies between two baselines, proving its efficacy in understanding and processing complex emotional nuances in text data.

The Decision Tree Classifier (best performing ML classifier), used as a lower-bound baseline, initially demonstrates a deceivingly competitive performance with F1-macro scores and accuracy that nearly approach those of more sophisticated models on the original dataset splits. However, when we experiment by switching the training and testing datasets—testing on data not seen during training—the Decision Tree's performance significantly degrades. This drop in performance underscores a critical limitation of traditional machine learning classifiers: they heavily depend on the nature of the training data and struggle to generalize to new, unseen contexts. This is particularly evident when testing the model trained on Dataset1 (Messages) on Dataset2 (Tweets), where the unique semantic structure and brevity of Tweets impact the model's ability to effectively parse and interpret emotional content.

Moreover, the testing results on unseen datasets highlight the adaptability of our model. While traditional classifiers exhibit dramatic performance reductions when confronted with new data types or shifted contexts, we maintain relatively stable and high performance. This robustness is a testament to the advanced capabilities of transformers-based models, which are designed to capture a deeper, more nuanced understanding of language context than their machine learning counterparts.

This experiment not only demonstrates the superior performance of our model in known contexts but also its potential to maintain high accuracy in unfamiliar settings, a vital characteristic for practical applications in diverse real-world scenarios. The results advocate for the use of advanced NLP techniques, such as pretrained LLMs (Mistraal here), in applications where adaptability and robustness across varied text formats and domains are crucial.

Table 1: Experimental Results on 90% Test Split

	DS1: Messages		DS2: Tweets	
	F1-macro	Accuracy	F1-Macro	Accuracy
Baseline (Decision Tree C.)	0.56	0.87	0.62	0.72
Zero-shot	0.74	0.85	0.59	0.64
3-shot	0.78	0.90	0.66	0.70
Baseline RoBERTa	0.80	0.91	0.75	0.78

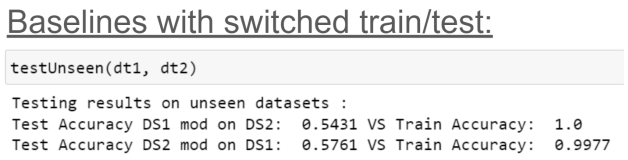


Figure 11: Switched Train vs Test Sets

6 Conclusion

Throughout this project, significant contributions were made towards enhancing natural language processing capabilities, specifically in emotional text analysis:

- Developed a codebase (gitHub Repo) that can run Mistral with user specific prompts, datasets, arguments and overall is quite modulable
- A set of accuracy efficient prompts
- A set of obtained results after 48 hours+ runtime
- This report

This project not only advanced my practical skills in areas such as GPU set-up, Pandas, PyTorch, leveraging Docker for deployment (tried having simulated Ubuntu), LateX (which I had never used) and ML, but also provided deep insights into effective data handling and GPU utilization for complex NLP tasks, and for that I must thank the entirely of the IDIAP lab, this was overall tremendously interesting although gratly time-consuming.

References

[1] *Dataset 1*. URL: https://www.kaggle.com/datasets/jarvis11/text-dataset-for-text-emotion-detection.

[2] *Dataset 2*. URL: https://github.com/SannketNikam/Emotion-Detection-in-Text/tree/main/data.

[3] *Project Repository*. URL: https://github.com/ahuou/SentimentAnalysis-PretrainedLLM.

[4] *Git Issue for Llama on Windows*. URL: https://github.com/abetlen/llama-cpp-python/issues/721#issuecomment-1723205068.

[5] *Prompt-Engineering course*. URL: https://www.deeplearning.ai/short-courses/prompt-engineering-with-llama-2/.