ISEN MÉDITERRANÉE | L'ÉCOLE DES INGÉNIEURS DU NUMÉRIQUE

GENERATIVE IA PROJECT REPORT

Members:

Layana FOUMO – Work Package Manager

Cassandra SOPP – Researcher

Aminata YADE – Data engineer

Rayan Freud FOUODJI TCHINDA – Integration engineer

Djibril DJOU KENNE – ML – Training

Hassania KABOULI – Quality Tester

Mr. Clément GICQUEL

Contents

Abstract

The present project addresses the pressing challenge of moderating harmful content on social media platforms by developing an AI-based system capable of detecting and blocking abusive messages. The targeted categories include hate speech, racism, homophobia, threats, insults, and other toxic forms of communication. Unlike conventional toxicity classifiers, our approach explicitly considers **contextual nuances**: sarcastic remarks, dark humor, or ironic statements are not automatically categorized as harmful, as they may lack malicious intent.

This topic was chosen due to its **high societal relevance** and **scientific complexity**. Online harassment and toxic interactions represent major threats to digital well-being, making effective moderation a priority for both industry and research. At the same time, the problem is technically demanding because toxicity is not only a matter of lexical content but also of pragmatics, tone, and conversational context. Existing systems often fail on these borderline cases, leading either to over-blocking (false positives) or under-blocking (false negatives).

The project aims to design and implement an **AI moderation pipeline** that combines multiple signals — toxicity classification, emotion and sentiment analysis, and sarcasm detection — to make more reliable moderation decisions. When a truly harmful message is detected, the system blocks it and issues a warning to the user. When the message is ambiguous, it issues a contextual warning rather than an outright block. In doing so, the system strikes a balance between **protecting users from abuse** and **preserving freedom of expression** in online environments.

Part I: RESEARCH

During the research phase, the focus was on surveying the state of the art in toxicity detection and identifying tools and datasets suitable for our project's objectives. Several key strands of work were completed:

1.1 Literature and tool review

The first step consisted of a targeted review of existing moderation systems and academic resources.

- **Perspective API (Google Jigsaw):** analyzed as a baseline service that provides toxicity and related attribute scores (e.g., insult, identity attack). While powerful, its black-box nature and lack of transparency limit its usefulness as the sole solution.
- **Jigsaw/Kaggle Toxic Comment datasets:** identified as the most widely used benchmark for toxicity classification, covering multiple subtypes (toxic, severe toxic, insult, threat, identity hate).
- **Sarcasm and context in toxicity detection:** reviewed studies showing that models often fail to differentiate between sarcastic humor and genuine abuse, highlighting the need for additional contextual signals.
- **Pretrained Hugging Face models:** explored two publicly available classifiers to complement toxicity prediction:
  - j-hartmann/emotion-english-distilroberta-base, which detects emotions such as anger, disgust, and joy.
  - cardiffnlp/twitter-roberta-base-sentiment, a sentiment classifier tuned on tweets.

These models were chosen because emotions like anger or disgust strongly correlate with abusive intent, while sentiment analysis provides additional polarity information. Together, they enrich the context around potentially toxic text.

1.2 Dataset exploration

Multiple datasets were identified and inspected in the provided materials:

- Small balanced sets: three CSV files (106 to 500 rows each) containing binary labels (toxic, non-toxic), already balanced 50/50.
- Large emotion/sarcasm set: nearly 4,000 rows labeled with fine-grained emotions (anger, disgust, fear, joy, neutral, sadness, surprise) and a sarcasm label.

The inspection confirmed that while the toxicity datasets are small, they are sufficient for prototyping and can be augmented with larger public datasets. The emotion/sarcasm dataset is especially relevant to the project's emphasis on contextual understanding.

1.3 Experimental exploration

Colab notebooks were used to test baseline models and pipelines:

- Transformers and Datasets libraries were installed and used to tokenize the text and fine-tune pretrained models.
- DistilRoBERTa and RoBERTa classifiers were prototyped on subsets of the data to validate feasibility.
- Emotion and sentiment models were loaded from Hugging Face to evaluate their predictions on sample toxic and non-toxic sentences.

Initial tests confirmed that emotion detection (e.g., high anger or disgust scores) often aligns with toxicity, while sarcasm-labeled data revealed misclassifications by basic toxicity models, validating the importance of multi-signal modeling.

1.4 Methodological choices

From these explorations, the following methodological decisions were made:

1. Multi-signal approach: toxicity detection alone is insufficient. The final architecture should combine toxicity classification with emotion and sentiment analysis and explicitly account for sarcasm.
2. Pipeline structure :
    - Stage 1: toxicity classifier (binary + subtype).
    - Stage 2: emotion and sentiment classifiers to contextualize predictions.
    - Stage 3: sarcasm detection to avoid false positives.
    - Stage 4: decision rules (block, warn, allow).
3. Evaluation strategy: metrics will include not only precision, recall, and F1 for toxicity, but also false positive rate on sarcastic non-abusive examples.

1.5 Outcomes of the research phase

The research phase delivered:

- A curated survey of existing APIs, datasets, and pretrained models.
- An assessment of the project's own datasets and their suitability.

- Prototype tests confirming the potential of Hugging Face models in detecting emotions and sentiment relevant to toxicity.
- A clear methodological direction: a context-aware moderation pipeline leveraging multiple classifiers.

1.6 Additional experimental insights

Beyond literature and exploratory analysis, two concrete models were tested during the research phase:

1. **A toxicity classifier**, which provided a simple toxic vs. non-toxic label. It worked in straightforward cases but failed when sarcasm or humor was present.
2. **An emotion classifier**, which predicted emotional states such as anger, joy, and disgust. This enriched the understanding of context and helped explain why some messages should or should not be flagged as toxic.

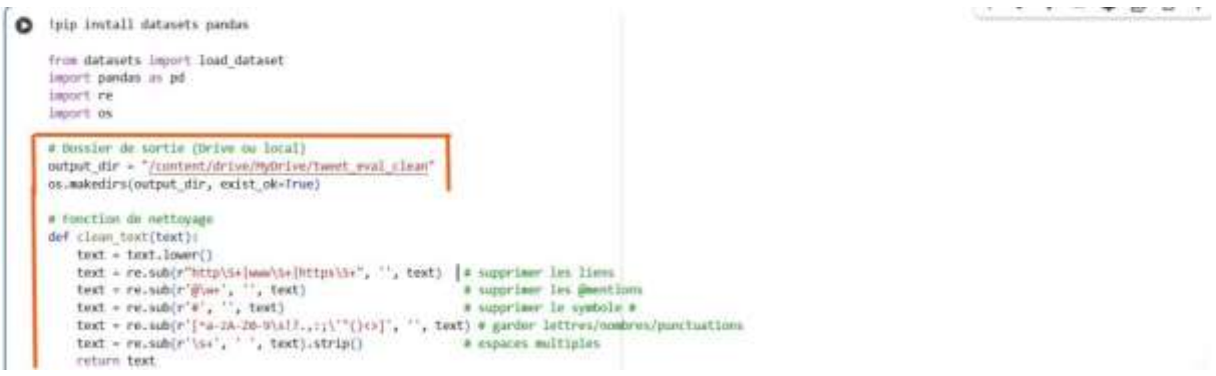
These two models demonstrated the practical challenges of toxicity detection: while binary classification is a starting point, emotion analysis is necessary to reduce false positives and better approximate human judgment.

Part II: DATA ENGINEERING

In this part, we describe the steps we performed during the data engineering phase. The goal was to design and prepare datasets suitable for our problem, ensuring consistency, reliability, and alignment with the contextual nature of toxicity detection.

2.1 First dataset attempt (Hugging Face)

The initial step was to test a dataset already available on Hugging Face. This dataset allowed us to quickly set up a pipeline and validate the data handling process. However, its labeling scheme was too generic and did not capture the contextual nuances of our project (e.g., sarcasm, humor). As a result, it was not retained for training.



```
!pip install datasets pandas

from datasets import load_dataset
import pandas as pd
import re
import os

# Dossier de sortie (Drive ou local)
output_dir = "/content/drive/MyDrive/tweet_eval_clean"
os.makedirs(output_dir, exist_ok=True)

# Fonction de nettoyage
def clean_text(text):
    text = text.lower()
    text = re.sub(r"http\S+|www\S+|https\S+", '', text)  # supprimer les liens
    text = re.sub(r'@\w+', '', text)                      # supprimer les @mentions
    text = re.sub(r'#', '', text)                         # supprimer le symbole #
    text = re.sub(r'[^a-zA-Z0-9\s!?.,:;\'"()<>]', '', text) # garder lettres/nombres/punctuations
    text = re.sub(r'\s+', ' ', text).strip()              # espaces multiples
    return text
```

2.2 Custom datasets creation

To adapt the data to our problem, we built our own datasets:

- **100-phrase dataset:** A small, balanced dataset (toxic vs. non-toxic) created to test feasibility. It provided an initial proof of concept.
- **500-phrase dataset:** An expanded dataset, created after the first tests, which allowed for more meaningful training and evaluation.

These custom datasets were critical because they directly reflected the type of messages we wanted our model to detect.

2.3 Archive datasets (additional sources)

Beyond the custom datasets, additional labeled files were found in the project archive:

- **Binary toxicity datasets:**
    - dataset (3).csv — 500 rows (250 toxic, 250 non-toxic).
    - dataset 1.csv — 500 rows (250 toxic, 250 non-toxic).
    - dataset.csv — 106 rows (53 toxic, 53 non-toxic).

- **Emotion and sarcasm datasets:**
    - examples.csv and examples (1).csv — 3,893 rows each, labeled with emotions (anger, disgust, fear, joy, neutral, sadness, surprise) and a sarcasm tag.

Together, these datasets provided both binary toxicity labels and context-related annotations for emotion and sarcasm.

2.4 Cleaning and harmonization

Several preprocessing steps were applied to ensure data consistency:

1. **Schema unification:** standardized fields (id, text, toxicity_label, emotion_label, sarcasm_label, source_file).
2. **Deduplication:** duplicates across datasets removed to prevent leakage.
3. **Normalization:**
    - UTF-8 encoding, whitespace and special characters cleaned.
    - URLs, emails, numbers replaced by placeholders ([URL], [EMAIL], [NUM]).
    - Emojis preserved for their contextual and emotional cues.
4. **Label consistency:** toxic/non-toxic mapped to binary (1/0); emotions and sarcasm labels verified.

2.5 Exploratory data analysis (EDA)

Exploratory analysis was conducted on the datasets:

- **Class balance:** binary datasets were balanced (50/50). Emotion datasets were skewed (anger = 730, disgust = 632, sarcasm = 403).
- **Text length:** average = 6 tokens, with 95% under 12 tokens → confirming that short-text models are suitable, with max_length=128.
- **Overlap:** sarcastic examples often overlapped with toxic labels, confirming sarcasm as a confounding factor.

2.6 Splitting strategy

Datasets were split for training and evaluation:

- **500-phrase dataset:** 75% training, 25% validation/test.
- **Binary archive datasets:** merged into 1,100 samples, split 70% / 15% / 15%.

- **Emotion/sarcasm datasets:** 70% / 15% / 15%, with proportional sarcasm representation.
- **Edge-case set:** 1,000 sarcastic/ambiguous examples reserved as holdout.

2.7 Feature preparation

In addition to raw text, several features were engineered:

- **Textual:** message length, capitalization, punctuation count.
- **Semantic:** profanity matches, emoji counts, hashtags/mentions.
- **Contextual:** placeholders for conversation metadata (if available).

2.8 Technical implementation

- **Libraries:** Hugging Face *Transformers* and *Datasets*.
- **Tokenization:** RoBERTa tokenizer, padded/truncated to 128 tokens.
- **Formats:** stored in CSV and Hugging Face Dataset format.
- **Reproducibility:** preprocessing pipelines implemented in Colab notebooks and exported as scripts.

2.9 Outcomes

The data engineering stage delivered:

- An initial test with a Hugging Face dataset, confirming the need for custom data.
- A **100-sample dataset** for feasibility, expanded to a **500-sample dataset** for training and evaluation.
- Cleaned, harmonized, and balanced archive datasets.
- A consistent schema and engineered features.
- Stratified train/validation/test splits and a dedicated edge-case holdout.

This work provided the foundation for the **ML Training phase**, where we tested Few-Shot prompting and fine-tuning approaches.

Part III: ML – Training

In this section, we describe the steps we performed during the ML training phase. The goal was to transform the prepared datasets into effective classification models and to compare different training approaches. Two methods were tested: **Few-Shot prompting** and **Fine-Tuning**.

3.1 First approach: Few-Shot prompting

The first training attempt relied on Few-Shot prompting. Instead of fine-tuning an entire model, a small number of examples were provided as prompts to guide the model's predictions.

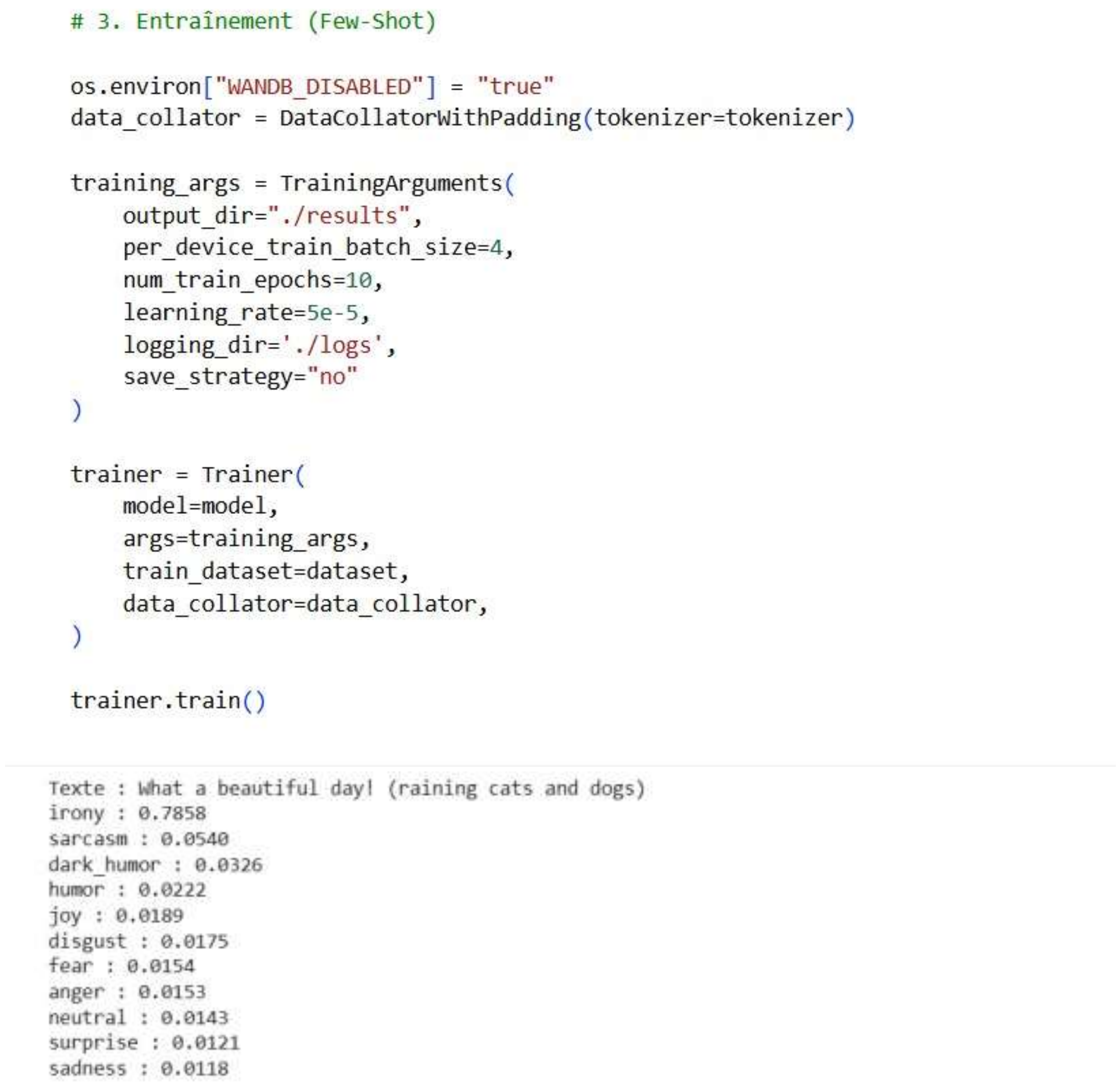
Advantages observed:

- Quick to set up and required no additional training.
- Allowed us to test model behavior immediately on toxic vs. non-toxic text.

Limitations identified:

- **Execution time:** classification required long inference times, as each query had to reprocess prompts.
- **Instability:** results were inconsistent across runs, with small changes in the prompt leading to very different outputs.
- **Scalability:** not suitable for large-scale deployment, as maintaining prompts for multiple cases became impractical.
- **Alignment with problem:** the method was not reliable enough to handle sarcasm, humor, or nuanced cases.

For these reasons, Few-Shot prompting was abandoned as a final solution, but it provided useful insight for moving forward.



```
# 3. Entraînement (Few-Shot)

os.environ["WANDB_DISABLED"] = "true"
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

training_args = TrainingArguments(
    output_dir="./results",
    per_device_train_batch_size=4,
    num_train_epochs=10,
    learning_rate=5e-5,
    logging_dir='./logs',
    save_strategy="no"
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=dataset,
    data_collator=data_collator,
)

trainer.train()
```

```
Texte : What a beautiful day! (raining cats and dogs)
irony : 0.7858
sarcasm : 0.0540
dark_humor : 0.0326
humor : 0.0222
joy : 0.0189
disgust : 0.0175
fear : 0.0154
anger : 0.0153
neutral : 0.0143
surprise : 0.0121
sadness : 0.0118
```

3.2 Second approach: Fine-Tuning

After the limitations of Few-Shot prompting were established, we adopted **Fine-Tuning** of pretrained models using our custom datasets.

- **Phase 1 – 100-phrase dataset:**
  The first fine-tuning experiments were conducted on the small 100-sample dataset. While this confirmed pipeline feasibility, the dataset was too limited to achieve reliable generalization.

- **Phase 2 – 500-phrase dataset:**
  The dataset was expanded to 500 samples for more meaningful training. Although still relatively small, it provided enough data to observe consistent learning patterns.

3.3 Training configuration

Fine-tuning was conducted in Google Colab with GPU acceleration.

- **Frameworks:** Hugging Face *Transformers* and *Datasets*.
- **Tokenizer:** RoBERTa-based, with truncation/padding to max_length=128.
- **Split:** 75% training, 25% validation/testing.
- **Optimizer:** AdamW with weight decay.
- **Loss function:** Binary Cross-Entropy for toxicity classification.
- **Epochs:** 3–5, with early stopping on validation loss.
- **Batch size:** tuned between 16 and 32, depending on GPU memory.

3.4 Results and observations

- **Improvement over Few-Shot:** fine-tuning produced more stable and consistent results.
- **Precision:** acceptable, meaning the model rarely mislabeled non-toxic comments as toxic.
- **Recall:** weaker, with some toxic comments missed, especially sarcastic ones.
- **Dataset size limitation:** both training phases (100 and 500 samples) were constrained by lack of data, preventing high robustness.

3.5 Outcomes of the training phase

The ML training stage delivered:

- A comparison of Few-Shot prompting and Fine-Tuning, demonstrating that fine-tuning is more effective.
- A trained toxicity classifier based on a 500-sample dataset, with measurable though limited performance.
- Insights into dataset requirements: to achieve production-level accuracy, thousands of labeled samples would be needed.

This work laid the foundation for the **Quality Assurance phase**, where model performance was evaluated more systematically.

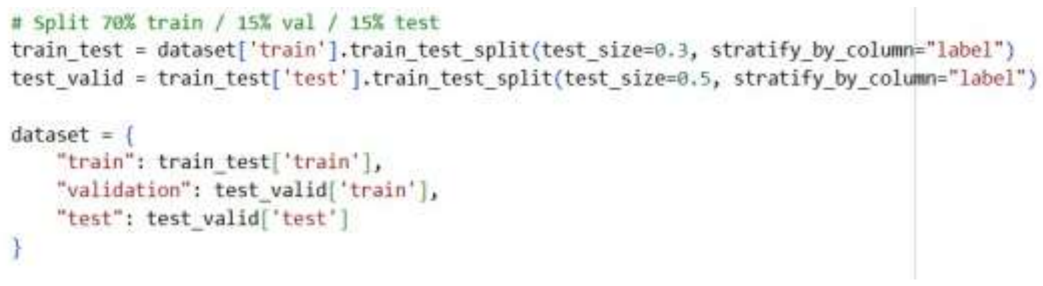
Part IV: QUALITY TEST

In this section, we describe the steps we performed during the quality evaluation phase. The objective was to systematically measure the performance of the trained models and verify whether they aligned with the project's requirements.

4.1 Evaluation dataset

The evaluation was conducted primarily on the **500-phrase custom dataset**.

- **Split** : 75% training, 25% validation/testing.
- **Stratification:** ensured an equal distribution of toxic and non-toxic examples across splits.
- **Data loading:** the dataset was loaded and preprocessed using Hugging Face *Datasets*. Standard tokenization (RoBERTa tokenizer, max_length=128) was applied consistently across both training and evaluation to ensure reproducibility.



```
# Split 70% train / 15% val / 15% test
train_test = dataset['train'].train_test_split(test_size=0.3, stratify_by_column="label")
test_valid = train_test['test'].train_test_split(test_size=0.5, stratify_by_column="label")

dataset = {
    "train": train_test['train'],
    "validation": test_valid['train'],
    "test": test_valid['test']
}
```

4.2 Evaluation metrics

The following metrics were used to evaluate performance:

- **Precision:** proportion of predicted toxic messages that were truly toxic.
- **Recall:** proportion of actual toxic messages that the model correctly identified.
- **F1-score:** harmonic mean of precision and recall, capturing balance between the two.
- **Confusion matrix:** to visualize misclassifications and identify systematic errors.

4.3 Results

- **Precision:** the model performed well, rarely labeling non-toxic comments as toxic.
- **Recall:** lower than desired, with several toxic messages being missed — especially sarcastic ones.
- **F1-score:** moderate, reflecting the imbalance between precision and recall.
- **Confusion matrix analysis:** showed that borderline cases (sarcasm, dark humor) were the most frequent source of errors.

4.4 Observations and limitations

- **Dataset size:** the 500-phrase dataset was too small to guarantee robustness. For reliable production use, thousands of labeled examples would be required.
- **Generalization:** while the model correctly classified straightforward toxic and non-toxic text, it struggled with nuanced cases where intent depended on humor or context.
- **Evaluation process:** the pipeline for loading, preprocessing, and validating data worked reliably, confirming the soundness of the methodology even though the data itself was limited.

4.5 Outcomes of the quality phase

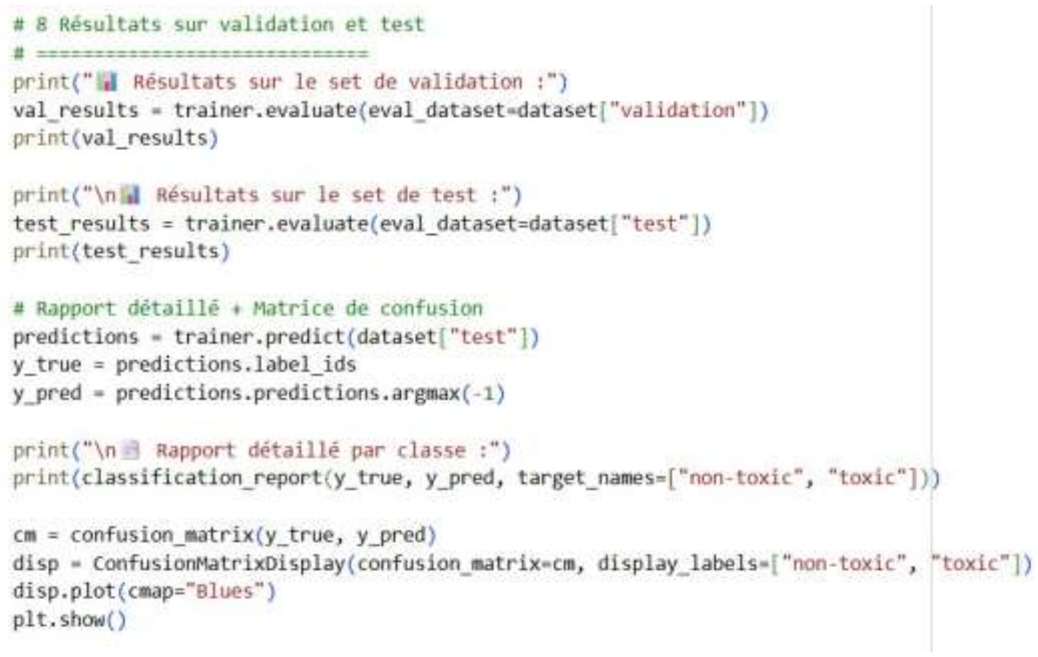
The quality evaluation phase delivered:

- A systematic performance assessment using precision, recall, F1, and confusion matrices.
- Confirmation that fine-tuning performs better than Few-Shot prompting.
- Identification of the main weakness: limited recall on sarcastic and context-heavy cases.
- Validation of the data handling and evaluation process in Colab, ensuring reproducibility.

These findings informed the **Integration phase**, where the trained model was connected to an interface and tested in a simulated real-world environment.

Part V: INTEGRATION

In this section, we describe the steps we performed during the integration phase. The goal was to connect the trained models into a functional moderation pipeline and expose it through a usable interface.



```python
# 8 Résultats sur validation et test
# ============================
print("📊 Résultats sur le set de validation :")
val_results = trainer.evaluate(eval_dataset=dataset["validation"])
print(val_results)

print("\n📊 Résultats sur le set de test :")
test_results = trainer.evaluate(eval_dataset=dataset["test"])
print(test_results)

# Rapport détaillé + Matrice de confusion
predictions = trainer.predict(dataset["test"])
y_true = predictions.label_ids
y_pred = predictions.predictions.argmax(-1)

print("\n📋 Rapport détaillé par classe :")
print(classification_report(y_true, y_pred, target_names=["non-toxic", "toxic"]))

cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["non-toxic", "toxic"])
disp.plot(cmap="Blues")
plt.show()
```

5.1 API implementation with FastAPI

The trained models were integrated into a **REST API** built with **FastAPI**, chosen for its efficiency and simplicity.

- **Endpoints:** the API provided an endpoint where a text message could be submitted via POST request.
- **Pipeline connection:** submitted messages were tokenized and processed by the toxicity and emotion classifiers.
- **Response schema:** the API returned structured outputs, including:
  - toxicity_label (toxic or non-toxic).
  - confidence (probability score for toxicity).
  - emotion (dominant emotion detected).
  - emotion_score (confidence score for the detected emotion).

This setup allowed users or external systems to interact with the moderation pipeline in a standardized way.

5.2 Decision rules

The decision logic implemented in the pipeline was based on model outputs:

- **If toxic** → the message is blocked, and the user receives a warning.
- **If non-toxic** → the message is accepted.
- **If ambiguous (e.g., sarcasm or dark humor)** → the message is allowed but a warning is displayed to the sender.

These rules ensured that the system avoided over-blocking while still preventing harmful content from being published.

5.3 User interface

In addition to the API, a **simple front-end interface** was developed for demonstration.

- Users could input text directly into the interface.
- The system displayed the classification result in real time, along with the additional metadata (confidence, emotion, emotion_score).
- This made the moderation process transparent, allowing users to understand why their message was flagged or blocked.

5.4 Technical setup

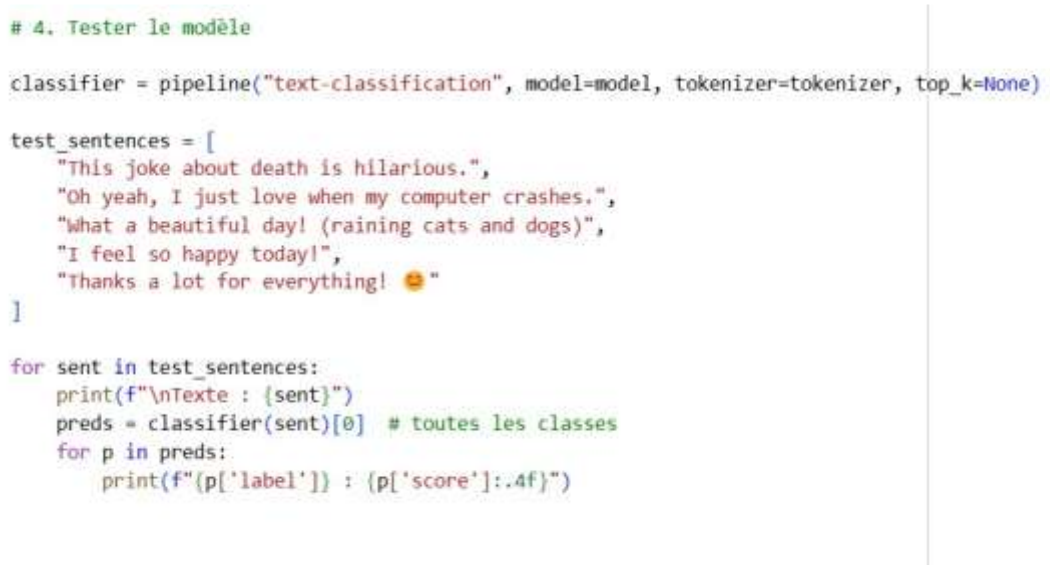
- **Frameworks:** FastAPI for backend, Hugging Face Transformers for model inference.
- **Deployment:** API tested locally through Colab notebooks and FastAPI's interactive Swagger UI.
- **Performance:** Inference was relatively fast on small datasets, but real-time deployment at scale would require optimization (e.g., batching, model quantization).

5.5 Outcomes of the integration phase

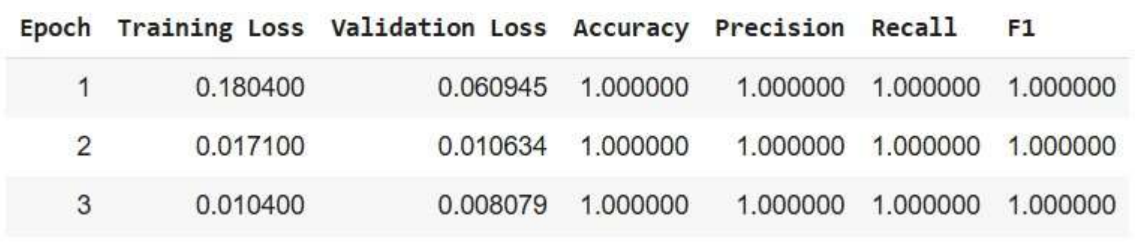
The integration stage delivered:

- A working moderation pipeline accessible via **REST API**.

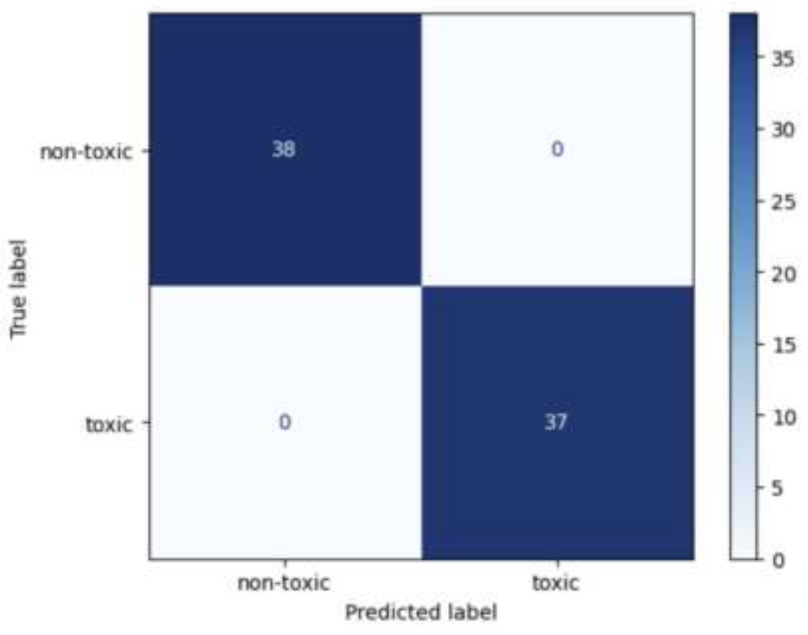
- Structured responses providing not only binary toxicity classification, but also contextual signals (emotion and confidence scores).
- A demo interface demonstrating how end-users would interact with the system.
- Proof of concept for a scalable moderation service, while also highlighting the need for further optimization and larger datasets for production readiness.



```python
# 4. Tester le modèle

classifier = pipeline("text-classification", model=model, tokenizer=tokenizer, top_k=None)

test_sentences = [
    "This joke about death is hilarious.",
    "Oh yeah, I just love when my computer crashes.",
    "What a beautiful day! (raining cats and dogs)",
    "I feel so happy today!",
    "Thanks a lot for everything! 😊"
]

for sent in test_sentences:
    print(f"\nTexte : {sent}")
    preds = classifier(sent)[0]  # toutes les classes
    for p in preds:
        print(f"{p['label']} : {p['score']:.4f}")
```



Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.180400	0.060945	1.000000	1.000000	1.000000	1.000000
2	0.017100	0.010634	1.000000	1.000000	1.000000	1.000000
3	0.010400	0.008079	1.000000	1.000000	1.000000	1.000000



CHALLENGES AND LIMITATIONS

Despite the progress achieved across the five phases of the project, several challenges and limitations were encountered. These issues highlighted both the complexity of toxicity detection and the need for future improvements.

1. Dataset size and representativeness

One of the main difficulties was the limited availability of high-quality, task-specific data.

- The first dataset created contained only 100 phrases, which was sufficient for pipeline validation but far too small for robust training.
- Even after expanding to 500 phrases, the dataset size remained insufficient to capture the full variability of toxic language in real-world contexts.
- The imbalance in some labels (e.g., sarcasm, certain emotions) also constrained generalization.

As a result, the trained models were prone to overfitting and lacked robustness when tested on edge cases.

2. Few-Shot prompting limitations

The initial approach using Few-Shot prompting proved difficult to scale.

- Execution was slow, as prompts had to be reprocessed for each prediction.
- Results were unstable, with small changes in the prompt leading to significantly different outputs.
- The method lacked reliability for nuanced cases such as sarcasm or dark humor.

This confirmed that Few-Shot prompting was not a sustainable strategy for deployment.

3. Fine-Tuning constraints

While fine-tuning provided more stable results, it introduced new challenges:

- Training was computationally expensive, even with small datasets.
- Execution times in Colab were long, requiring careful resource management.
- The limited dataset size restricted the model's ability to generalize beyond the specific phrases it was trained on.

These factors showed that fine-tuning was the right direction, but only viable with larger datasets and more powerful compute resources.

4. Sarcasm and contextual understanding

Sarcasm and humor emerged as one of the hardest challenges.

- Toxicity classifiers alone misclassified sarcastic jokes as harmful.
- Emotion classifiers added contextual signals (e.g., anger, disgust), but sarcasm remained a source of false positives and false negatives.

This highlighted the need for more advanced modeling approaches, possibly including multi-modal data or conversation context.

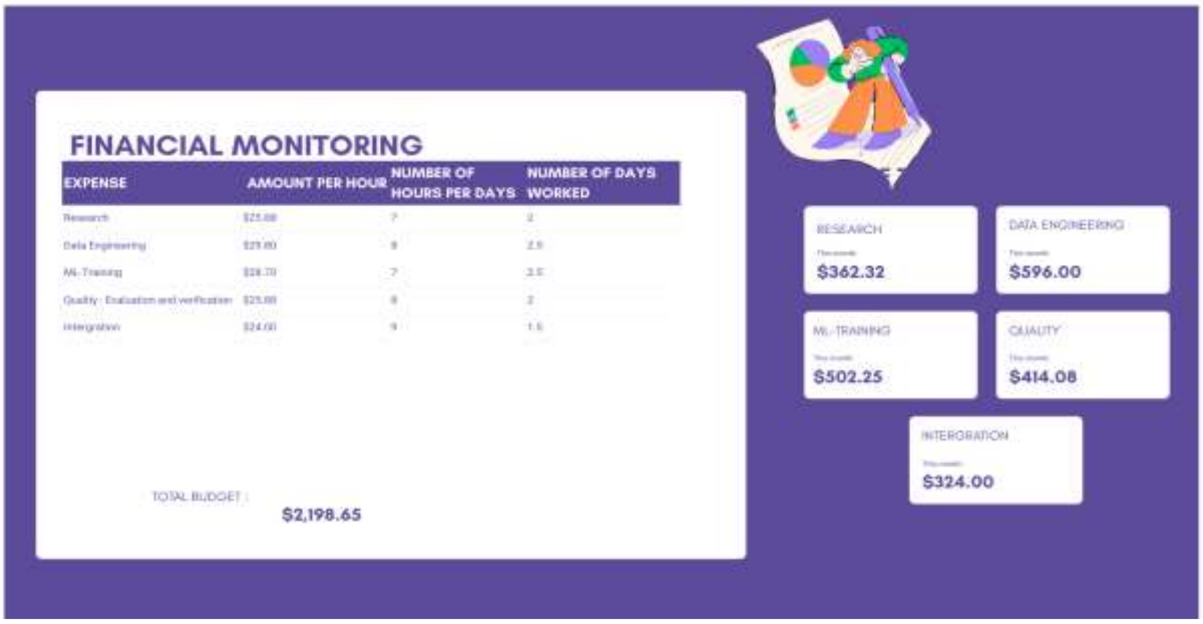
5. Integration and scalability

The integration of the models into a Fast API-based REST service worked as a proof of concept, but also revealed practical challenges:

- The system was functional for small-scale testing but would need optimization (e.g., model compression, batching, caching) for real-time production use.
- Latency could become problematic in high-traffic environments.

- The API outputs (confidence, emotion, emotion_score) improved interpretability, but user acceptance of moderation decisions remains a broader challenge.

FINANCIAL AND PHYSICAL MONITORING

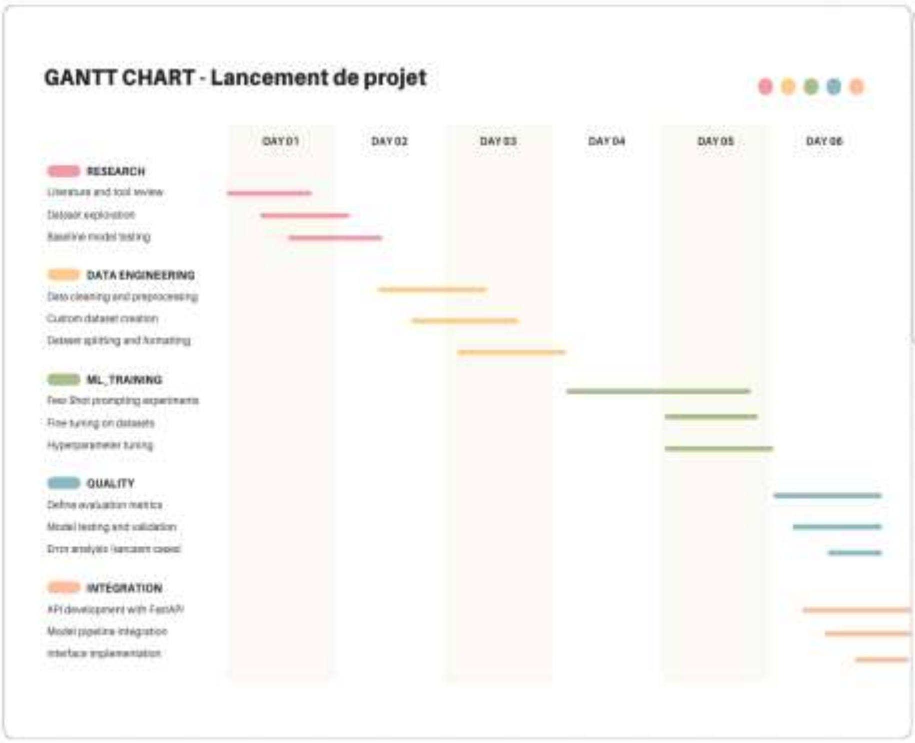


FINANCIAL MONITORING

EXPENSE	AMOUNT PER HOUR	NUMBER OF HOURS PER DAYS	NUMBER OF DAYS WORKED
Research	$25.88	7	2
Data Engineering	$29.80	8	2.5
ML-Training	$28.70	7	2.5
Quality : Evaluation and verification	$25.88	8	2
Intergration	$24.00	9	1.5

TOTAL BUDGET : **$2,198.65**

RESEARCH	**$362.32**
DATA ENGINEERING	**$596.00**
ML-TRAINING	**$502.25**
QUALITY	**$414.08**
INTEGRATION	**$324.00**



TEAM

- LF Layana FOUMO - Work Package Manager
- CS Cassandra SOPP - Researcher
- AY Aminata YADE - Data Engineer
- DK Djibril DJOU KENNE - ML_Training engineer
- HK Hassania KABOULI - Quality inspector
- RT Rayan Freud FOUODJI TCHINDA - Integration