Classic ML Models for Text Classification Offensive/Not Offesive Instagram Comments Meta 2 Project

Pedro Silva, Ramyad Raadi
uc2023235452@student.uc.pt, uc2023205634@student.uc.pt

University of Coimbra, Faculty of Science and Technologies
Bachelor of Artificial Intelligence and Data Science, PLN Course

November 2025

1 Introduction

In the initial phase of this project (Meta 1), we developed a rule-based classification system to identify offensive comments. This approach achieved an overall accuracy of 78%. The results demonstrated that this rule-based method was effective (and interpretable) for the task.

This new study shift the focus to the development of a classification system using traditional machine learning methods. The primary objective is to explore whether these models can improve upon the 78% accuracy baseline established by our rule-based system.

1.1 Preprocessing and Lemmatization

We adopted the core preprocessing pipeline from Meta 1, which includes removing punctuation, normalizing text, and strategically retaining emojis, as it proved to be a valuable feature.

Based on feedback from Meta 1, we re-evaluated lemmatization. While this new test yielded more consistent and stable results than our initial attempts, it still led to a slight decrease in overall performance metrics. Therefore, lemmatization was definitively excluded from our final pipeline in favor of using the original token forms in meta 1.

2 Methodology

Our methodology was structured into a multi-phase experiment, building upon the preprocessing pipeline developed in Meta 1. The dataset was split into 80%

for training and 20% for testing.

2.1 Phase 1: Best Vectorizer Method

The first phase aimed to find the optimal feature representation for the HateBR dataset. We used `LogisticRegression` as a fixed control model to ensure a consistent baseline and used train-test cross validation for evaluation.

- **Methods Tested:** We evaluated four distinct vectorization methods: `CountVectorizer`, `TfidfVectorizer`, `Word2Vec` (using averaging), and `Doc2Vec`.
- **Lemmatization Test:** Crucially, we tested all four methods both *with* and *without* lemmatization to definitively measure its impact.
- **Robustness:** Each of the 8 combinations (4 methods x 2 preprocessing) was run multiple times (`N_RUNS = 5`), and the F1, Precision, and Recall scores were averaged to ensure the stability and reliability of the results.

2.1.1 Results from Phase 1

From this phase, we identified the single best-performing representation for use in the next phase: **CountVectorizer**, as observed in table 1 and 2.
Also, we could yield several key insights:

- **Sparse vs Dense:** For classic models, sparse representations (`CountVectorizer` / `TfidfVectorizer`), which focus on *which* words were said, clearly outperformed dense representations (embeddings), which focus on context.
- **Lemmatization vs No Lemmatization:** Lemmatization provided a small but consistent performance boost for the winning models. This proves that for this problem, grouping related words (e.g., "doente" and "doentes") was beneficial.
- **CountVectorizer vs TfidfVectorizer:** Curiously, `CountVectorizer` (simple word count) was slightly better than `TfidfVectorizer`. This suggests that for hate speech, the mere presence of an offensive word may be more important than its rarity across the entire corpus (which TF-IDF measures).

2.2 Phase 2: Model Optimization with Optuna

The second phase focused on finding the best hyperparameters for a set of top classic models, using the best text representation identified in Phase 1 and used a K-Fold cross validation for evaluation (10 folds).

- **Models Tested:** We selected three classifiers for tuning: `LogisticRegression`, `LinearSVC` (Linear Kernel SVM), and `MultinomialNB`.

Vectorizer	Lemmat.	Mean F1	Std F1	Mean Precision	Mean Recall
CountVectorizer	**True**	**0.8415**	**0.0117**	**0.8447**	**0.8419**
TfidfVectorizer	True	0.8344	0.0111	0.8347	0.8344
CountVectorizer	False	0.8340	0.0106	0.8382	0.8344
TfidfVectorizer	False	0.8288	0.0059	0.8293	0.8289
Doc2Vec	True	0.7689	0.0067	0.7696	0.7690
Doc2Vec	False	0.7594	0.0058	0.7597	0.7594
Word2Vec	True	0.7309	0.0100	0.7417	0.7333
Word2Vec	False	0.7293	0.0107	0.7444	0.7326

Table 1: Phase 1 Results (Ranked by Mean F1)

Metric	Value
Vectorizer	CountVectorizer
Lemmatization	True
Mean F1	0.841522
Std F1	0.011703
Mean Precision	0.844723
Mean Recall	0.841857

Table 2: Best Combination Details

- **Model Justification:** This selection was deliberate. `LinearSVC` and `LogisticRegression` are widely considered state-of-the-art for classic text classification. They are linear models that are well known at finding patterns in high-dimensional, sparse feature spaces. `MultinomialNB` was included as a fast and powerful baseline, which is highly effective despite its "naive" assumption of word independence. These conclusions were retrieved from [2] and [3].
- **Tuning Framework:** We utilized the `Optuna` framework for efficient and powerful hyperparameter optimization.

2.2.1 Results from Phase 2

Using the `Optuna` framework, we tuned all three models by maximizing the weighted F1-score. The `LogisticRegression` model emerged as the clear winner, achieving the best balance of Precision, Recall, and, critically, the highest F1-score.

The `LinearSVC` model also performed exceptionally well, closely matching the `LogisticRegression`, which confirms that linear models are the correct choice for this task. `MultinomialNB` provided a strong baseline but could not match the performance of the other two models. The best hyperparameters for the winning `LogisticRegression` model are detailed in Tables 3 and 4.

Metric	Value
Best F1	0.8452
Best Precision	0.8488
Best Recall	0.8455

Table 3: Overall Winner: LogisticRegression Metrics

Hyperparameter	Value
vect_ngram_range	(1, 1)
vect_min_df	1
vect_max_df	0.79809088
clf_solver	'saga'
clf_penalty	'l2'
clf_C	10.66314227

Table 4: Overall Winner: Best Hyperparameters

3 Results and Discussion

The experiments in Meta 2 yielded a significant improvement over the Meta 1 rule-based system. The final tuned `LogisticRegression` model achieved a final f1 score of **85%**, a 7-point increase from the 78% f1 score of the Meta 1 system.

While the overall metrics improvement is notable, the primary success of the machine learning model is revealed in the confusion matrix (Figure 2) and its comparison to the Meta 1 results (which had 230 False Negatives.

The final classification report (Figure 1) shows a robust and balanced model, achieving a macro F1-score of 0.85, performing almost identically for both classes.

The most critical improvement was the reduction of **False Negatives (FNs)**, which are offensive comments incorrectly labeled as "Not Offensive."

- **Meta 1 (Rule-Based):** Produced **230 FNs**.
- **Meta 2 (LogisticRegression):** Produced only **125 FNs**.

This represents a **45.6% reduction in FNs**, proving the machine learning approach is far superior for the main task of detecting offensive speech. This was achieved with almost no change in False Positives (80 vs. 83), making it a highly beneficial trade-off.

	precision	recall	f1-score	support
Not Offensive (0)	0.83	0.88	0.86	700
Offensive (1)	0.87	0.82	0.85	700
accuracy			0.85	1400
macro avg	0.85	0.85	0.85	1400
weighted avg	0.85	0.85	0.85	1400

Figure 1: Classification report of the best experiment

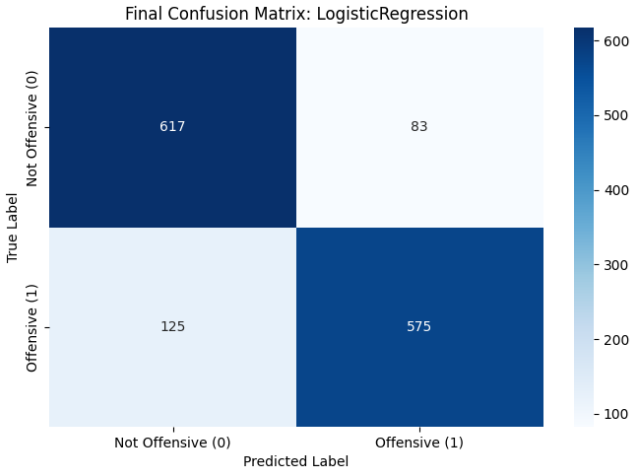

Figure 2: Confusion matrix of the best experiment

4 Acknowledgments

We would like to mention that AI [Gemini] was used to:

- Correct some syntax errors and make the report more cohesive and coherent;
- Provide insights on the methodology approach;
- Building some components of the code.

5 References

1. Hugo Oliveira, Isabel Carvalho and Patrícia Ferreira's slides and documents
2. Joachims, T. (1998). *Text categorization with support vector machines: Learning with many relevant features.* In Proceedings of the 10th European Conference on Machine Learning (ECML-98), pp. 137–142.
3. Wang, S., & Manning, C. D. (2012). *Baselines and bigrams: Simple, good sentiment and topic classification.* In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 90–94.
4. Optuna Framework: `https://www.kaggle.com/code/stpeteishii/emoji-with-u200d`
5. tqdm Framework for visual completion bars: `https://tqdm.github.io/`