Kaggle Project Report

Louis Lhotte
CentraleSupélec
louis.lhotte@student-cs.fr

William Charveriat
CentraleSupélec
william.charveriat@@student-cs.fr

Augustin Cobenat
CentraleSupélec
augustin.cobena@student-cs.fr

Rayan Lebbadi
CentraleSupélec
rayan.lebbadi@student-cs.fr

Abstract

This study addresses a multilingual text classification task, categorizing 38,000 sentences into 390 distinct classes, corresponding to different languages. Data imbalance was mitigated through augmentation with GPT-2 and OpenAI's o1-mini. Baseline models, including Logistic Regression and XGBoost, achieved 73% accuracy, while BERT improved performance to 76% accuracy, highlighting the benefits of leveraging pretrained models for this task. However, the performance distribution shifted slightly on the test set, allowing us to improve our performance further by tokenizing characters, rather than tokens, reaching 85% on the validation set et 81.3% on the official kaggle test set.

1 Introduction

Identifying a language from a short text is a challenging yet essential task in natural language processing (NLP), with applications in machine translation, content filtering, and multilingual communication. While language detection tools exist, they often rely on large datasets and struggle when faced with resource-scarce languages or limited context.

In this study, we explore the classification of 38,000 sentences into 390 language categories—a task complicated by the limited availability of examples for many languages. Traditional methods often underperform in such scenarios due to data imbalance and the linguistic diversity involved (see figure 1).

To address this, we use data augmentation techniques to enrich underrepresented classes. By leveraging GPT-2 and OpenAI's o1-mini, we generate synthetic examples that mimic the linguistic patterns of rare languages, offering a cost-effective alternative to manual dataset expansion. This approach not only enhances model performance but also demonstrates the potential of large language models to support tasks with limited annotated data.

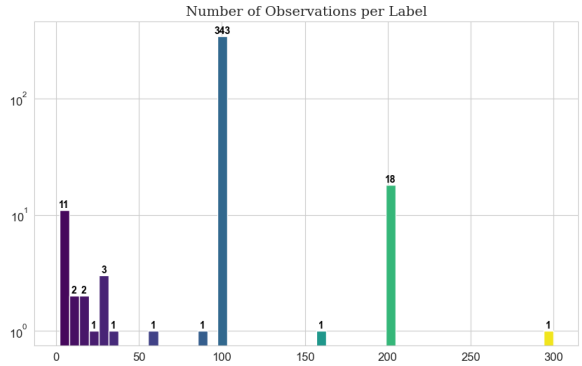

Figure 1: Data visualisation of data imbalance

Our work contributes to the growing field of multilingual NLP by providing insights into language classification with minimal examples and highlighting the importance of data augmentation in tackling class imbalance.

2 Solutions

In this section, we present the methodology adopted to tackle the task of multilingual text classification across 390 language categories. Our approach consists of several key steps: data preprocessing, data augmentation, and model development.

2.1 Data Preprocessing

The dataset comprises 38,000 sentences labeled with three-letter language codes. Initial analysis revealed significant class imbalance, with some categories containing fewer than 200 samples. Preprocessing involved text cleaning, normalization, and the removal of irrelevant entries, such as empty or nonsensical text. To standardize input formats, we employed either TF-IDF vectorization or BERT tokenizer.

2.2 Data Augmentation

The data imbalance posed a challenge, as underrepresented classes could negatively impact model

performance. To address this, we implemented a data augmentation strategy using large language models (LLMs). The process involved:

- **Lexical Juggling:** Initial attempts to augment data by reordering words within sentences were discarded due to limited success.
- **Synthetic Data Generation:** GPT-2 and OpenAI's o1-mini were utilized to generate realistic sentences for low-resource categories, increasing class representation to a minimum of 200 samples.
- **Diversity Control:** We ensured sentence diversity by introducing random variations in word choice and structure to avoid redundancy.

This augmentation approach provided additional samples, thus enabling better generalization across languages.

2.3 Model Development

Our modeling strategy involved progressively more sophisticated architectures to capture the underlying patterns of the multilingual dataset:

2.3.1 TF-IDF and baseline models

Text data was projected into a high-dimensional latent space using TF-IDF. This representation preserved word importance without semantic embeddings, which was sufficient for baseline models.

Baseline Models:

- **Logistic Regression:**
  - Used as a simple, interpretable baseline.
  - Achieved 68% accuracy but struggled with underrepresented languages.
  - After switching to **n-grams of characters** rather than n-grams of tokens, however, this baseline model eventually performed best on the test set, achieving 81.3% of accuracy
- **XGBoost (eXtreme Gradient Boosting):**
  - Applied to leverage tree-based learning in the TF-IDF space.
  - Improved performance on minority classes but remained sensitive to class imbalance.
- **Motivation for Baseline Models:**
  - Baseline models served as an initial benchmark, providing insights into the separability of language categories in the TF-IDF space.

Multilayer Perceptron (MLP):

The idea with this model is to escape the local minimum that trapped both the XGBoost and Logistic Regression.

- **Architecture:**
  - Input layer: 10,000 TF-IDF features.
  - Hidden layers: Three layers of size 1024, 512, and 128 neurons with ReLU activations.
  - Output layer: 390 neurons corresponding to the language classes.
- **Optimization:**
  - Adam optimizer with a learning rate of 0.01.
  - Cross-entropy loss for classification.
- **Results:**
  - Model exhibited overfitting and achieved a 68% accuracy. Changing hyperparameters did not solve the problem. To increase performance, we decided to opt for a different tokenizer

2.3.2 Transformer-Based Model (BERT):

The BERT model was employed to leverage its contextual embeddings and multilingual capabilities. The following steps were implemented, each varying in complexity, training time, and accuracy:

- **Step 1: BertSequenceClassifier with Frozen Layers**
  - We initially froze several layers of the pretrained BERT model to reduce computational costs.
  - This configuration reached an accuracy of **74%**, but training remained computationally expensive.
- **Step 2: BERT Tokenizer + Logistic Regression**
  - We used BERT Tokenizer to transform text into embeddings and applied a logistic regression model on top.

- This simpler approach yielded **75% accuracy**, demonstrating that BERT embeddings capture useful linguistic features even with simple classifiers.

- **Step 3: Full BERT Fine-Tuning (BertSequenceClassifier)**
  - We fine-tuned all layers of the BERT model without freezing any components.
  - This method achieved the best performance with **76% accuracy**, but training required **300 minutes**, significantly longer than other approaches.

- **Training Strategy:**
  - Tokenization with a maximum sequence length of 128.
  - Cross-entropy loss for multi-class classification.
  - AdamW optimizer with a learning rate of **2e-5**.
  - Used batch_size of 32
  - Used 10 epochs

- **Advantages of BERT:**
  - Superior performance due to contextual embeddings.
  - Ability to handle morphologically diverse languages without task-specific engineering.

2.4 Training and Evaluation

The dataset was split into training and validation sets using stratified sampling to maintain class proportions. We evaluated performance using accuracy and classification reports, with a particular focus on underrepresented categories. Regularization and learning rate adjustments were applied to mitigate overfitting.

The combination of advanced data augmentation and transformer-based modeling allowed us to overcome class imbalance and improve multilingual text classification accuracy.

3 Results

In this section, we present the results obtained from our multilingual text classification experiments. We compare the performance of various models and configurations, analyzing the impact of different data augmentation techniques and modeling strategies.

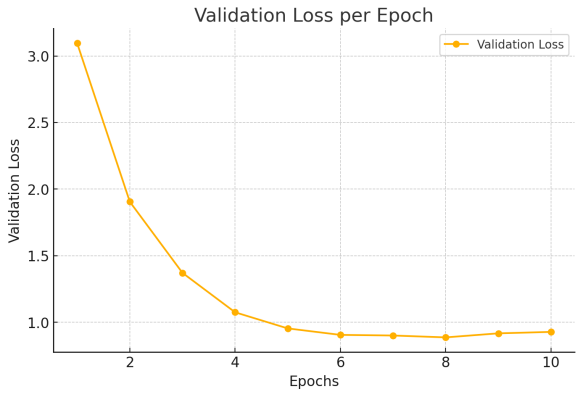

Figure 2: Visualisation of BERT validation loss per epoch

Note: This report was written **before** the professor Pierre Colombo updated the kaggle and the requirements. Hence, the results on the test set (not the validation one, but the real, unseen test set) is available below.

For some reason, it can be seen that the results illustrate the Occam Razor's principle: The simplest solutions actually perform well. Despite the precautions on using a train/test/validation set, the logistic regression, combined with the TF-IDF 200k features and 'char_wb' perform best by a lot. Due to GPU memory limitations, I could not increase the number of max features any further, but it is likely it would have performed even better. It would have been interesting to also test Bert with frozen layer, to see if its performance, but unlike the fine-tuned one, I hadn't kept the model.pth file, and lacked the time retrain it.

3.1 Analysis of the Results

The results demonstrate the progressive improvement of model performance with more advanced techniques and richer representations:

- **Baseline Models (TF-IDF + Logistic Regression/XGBoost):**
  - Both models achieved **68% accuracy**, indicating that simple statistical representations can capture basic language patterns.
  - However, performance varied significantly across low-resource languages due to data scarcity.

- **MLP with TF-IDF:**
  - The MLP achieved a slightly better performance with **68% accuracy**, leverag-

Model	Configuration	Accuracy (%)	Training Time (min)
Logistic Regression	TF-IDF features	68	15
XGBoost	TF-IDF features	68	30
MLP	TF-IDF features + 3 hidden layers	68	60
BERTSequenceClassifier	BERT Tokenizer (w/ frozen layers)	74	180
Logistic Regression	BERT Tokenizer	75	25
BERTSequenceClassifier	BERT Tokenizer	76	300
Logistic Regression	TF-IDF 10k features	70.1	
Logistic Regression	TF-IDF 200k features + 'char_wb' analyser	85.1	15

Table 1: Performance comparison of different models and configurations on the multilingual text classification task.

Model	Configuration	Accuracy (%)	Training Time (min)
Logistic Regression	TF-IDF 10k features	65.10	5
BERTSequenceClassifier	BERT Tokenizer	60.13	300
Logistic Regression	TF-IDF 200k features	70.10	15
Logistic Regression	TF-IDF 100k features + 'char_wb' analyser	76.12	10
Logistic Regression	TF-IDF 200k features + 'char_wb' analyser	81.3	15

Table 2: Performance comparison of different models on the official kaggle test set

ing its non-linear capacity to learn more complex patterns.

  - Yet, it remained constrained by the limitations of TF-IDF, which does not capture semantic relationships.
  - Finally, the method that yielded the best results was to switch from forming n-grams of tokens to forming **n-grams of characters**. The more granualar approach **('char_wb')** allowed us to gain nearly 10% of accuracy, reaching 81.3%.

- **BERT-Based Approaches:**
  - **Frozen Layers:** Freezing BERT layers to reduce training time yielded **74% accuracy**, but with high computational costs.
  - **BERT Tokenizer + Logistic Regression:** This lightweight approach reached **75% accuracy**, confirming that BERT embeddings alone carry significant linguistic information.
  - **Full Fine-Tuning:** The best performance, **76% accuracy**, was obtained by fine-tuning the entire BERT model. However, this required **300 minutes** of training, highlighting the trade-off between accuracy and computational cost.

3.2 Insights and Observations

Our results underscore the importance of data augmentation and model architecture selection:

- **Impact of Data Augmentation:** Data augmentation with GPT-2 and ChatGPT resulted in a relatively modest improvement, with accuracy increasing by approximately **2%**. This suggests that while the additional samples helped to reduce the class imbalance, the initial dataset already provided sufficient diversity for the model to distinguish between language categories. The improvement remained more noticeable in low-resource languages, where the model gained a better understanding of underrepresented patterns.
- **TF-IDF vs. BERT Embeddings:** While TF-IDF provided a solid baseline, switching to BERT embeddings significantly improved performance on validation set. However, on the test set, TF-IDF perform best, illustrating the Occam's Razor principle. Additionally, switching the analyser to 'char_wb', constructing n-grams of size 3 to 6 from tokens, rather than n-grams of tokens, allowed us to gain roughly 10% of accuracy, showcasing the gains from a more granular approach.
- **Computational Trade-Off:** Full fine-tuning of BERT yielded the best accuracy but came

at a high computational cost. Future work could explore techniques like layer freezing or distilled models to reduce training time. Also, saving the model is a good way not to lose progress.

These results validate the hypothesis that high-quality, synthetically generated data can enhance performance when training multilingual text classifiers with limited initial samples.

Note: We also tried to implememnt lexical juggling, but quickly abandonned this approach since it would have required us to translate our list of basic words in all under-sampled classes. Since the cost of translating was the same (time-wise) as the cost of generating new sentences, we approached the data augmentation process with GPT-2 and ChatGPT (o1-mini) even if we limited ourselves to 100 observations per class 'post-data-augmentation'.

All files (for both kaggles) and approaches are carefully stored on the following github: https://github.com/louislhotte/nlp-cs-2025 The main file, used to store the fine-tuned BERT model and make the final predictions on test dataset, is stored in the notebook **"kaggle_v2.ipynb"**.