Topic 49 - Language identification

Stoicescu Adrian Nicolae and **Tantaru Dragos Constantin**
Group No. 507

Abstract

This project addresses the challenging task of distinguishing between the Romanian and Moldovan dialects within textual datasets. Given the close linguistic similarities between the two dialects, standard language detection models often struggle to differentiate effectively. Our research focuses on the development and comparison of various supervised learning approaches to enhance the accuracy of dialect identification. In the documentation, we detail the methodology, experiments, and comparative analysis of the employed supervised learning models, highlighting their strengths, limitations, and potential areas for further research.

1 Dataset

The MOROCO (Moldavian and Romanian Dialectal Corpus) dataset is a specialized corpus created for analyzing and distinguishing Romanian and Moldavian dialects. This dataset, created by Andrei M. Butnaru and Radu Tudor Ionescu in 2018, is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Licence, which makes it possible for the dataset to be shared and adapted as long as appropriate credit is given and it is not used for commercial purposes.

MOROCO is designed to provide for a variety of classification tasks, including binary classification by dialect, multi-class classification by subject within each dialect, and cross-dialect classification by topic. The dataset contains 31,564 text samples, evenly divided across six categories: culture, money, politics, science, sports, and technology. The samples are separated into two subsets: training (21,719 samples) and testing (5,924 samples).

2 Features

For feature extraction from the sentences and generating feature vectors, we implemented two distinct methods. We additionally also used a finetuned LLM for the dialect task.

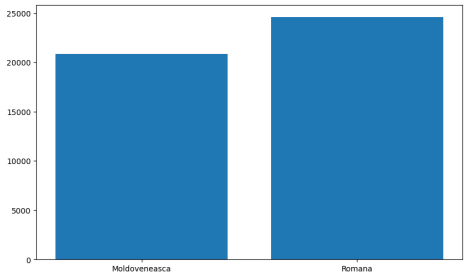

Figure 1: Visualization of the dialect distribution in the dataset

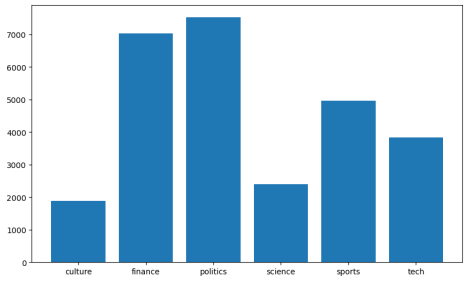

Figure 2: Visualization of the categories distribution in the dataset

2.1 TF-IDF

TF-IDF, which stands for Term Frequency-Inverse Document Frequency, is a statistical measure used to assess the significance of a term in a document in relation to a collection or corpus of documents. This approach is commonly used in the disciplines of natural language processing. The TF-IDF value rises with the frequency of a word's recurrence in a given document; however, this rise is offset by the term's prevalence throughout the corpus. This change is critical since it reduces the impact of terms that naturally appear more frequently, ensuring that the importance assigned to each word more properly reflects its true relation to the document's content.

2.2 Embeddings Model

We used text embeddings to extract characteristics from textual data using neural models all of them available through Hugging Face. Our first choice was the 'all-mpnet-base-v2' model, which is suggested by the Sentence Transformers library due to its excellent efficiency and embedding generation optimization. We then used the RoBERT model to improve our feature extraction method, particularly because we were trying to discern between Romanian and Moldovan dialects. This model was chosen precisely because it was trained on Romanian language data, which may provide an edge in catching subtleties that distinguish the two dialects more effectively.

3 Classifiers

In the field of supervised machine learning, solving classification problems is critical for anticipating discrete outputs from given input data. This study incorporates a diverse range of categorization systems, each defined by its style of analysis and prediction. These models were strategically chosen, with a focus on flexibility, ease of comprehension, and robustness when dealing with complicated datasets. These datasets frequently bring obstacles such as varying amounts of inter-feature correlations and concerns with unequal data distribution.

3.1 Decision Tree Classifier

A Decision Tree Classifier builds a prediction model in the format of a hierarchical, tree-like structure. It works by gradually splitting the data into smaller and smaller subgroups, while also constructing a tree with decision nodes and leaf nodes. Nonetheless, the model is prone to over-fitting, especially when working with complex datasets, which might result in less trustworthy predictions for fresh, unexplored data. Over-fitting happens when the model gets overly complicated, collecting noise in the data rather than the underlying pattern, resulting in poor performance on data outside of the training set.

3.2 Random Forest Classifier

During training, the Random Forest Classifier utilizes an ensemble learning technique that creates many decision trees and calculates the result based on the average predictions of these individual trees. This strategy improves accuracy and effectively reduces the danger of over-fitting by combining predictions from numerous deep decision trees, each trained on a distinct section of the dataset.

The standard Decision Tree methodology is considerably improved when using the Random Forest approach. This is accomplished by assembling a collection of decision trees, each created from a random subset of the dataset. This ensemble strategy not only tackles the over-fitting problem endemic to single Decision Trees, but it also frequently results in higher predicted accuracy. This enhancement is due to the collaborative decision-making process of several trees, which results in a more stable and generalized model capable of producing trustworthy predictions over a wide range of data circumstances.

3.3 Support Vector Classifier

The Support Vector Classifier (SVC) is a crucial component of Support Vector Machines (SVMs), designed to discover the best hyperplane for successfully separating distinct classes in a given feature space. This ideal hyperplane is distinguished by the greatest margin between the nearest points of the classes, known as support vectors. SVC is flexible and able to handle both linear classification jobs and complicated non-linear issues. The latter uses kernel functions to convert the input data into a higher-dimensional space, making it easier to separate the classes.

SVC excels in difficult classification problems when the data points are not linearly separable in their original space. It is renowned for its accuracy and ability to define complex decision boundaries. However, attaining these outcomes frequently necessitates fine-tuning of the settings. Furthermore, SVC can be resource-intensive, especially with huge datasets that require substantial processing capacity to handle.

3.4 K-Neighbors Classifier

The K-Neighbors Classifier belongs to the family of instance-based or "lazy" learning algorithms, which operate on the premise that similar instances tend to cluster together in the data space. This method identifies the 'k' nearest neighbors in the feature space for a given test instance and bases its prediction on the majority class among these neighbors. The choice of 'k' is crucial, as it influences the model's ability to generalize: a lower 'k' value increases sensitivity to data noise, while a higher 'k' enhances robustness against outliers,

albeit at the risk of reducing precision. This approach is inherently simple and can be highly effective, particularly in scenarios with complex decision boundaries. However, its efficacy may decline in high-dimensional spaces due to the curse of dimensionality, which complicates the identification of truly nearest neighbors due to the increase in "empty" space.

3.5 Logistic Regression

Despite its misleading name, logistic regression is fundamentally a linear method used primarily for binary classification, utilizing a logistic function to calculate probabilities. It is particularly adept at handling situations where the outcome can only be one of two distinct categories. Renowned for its simplicity and efficiency in binary contexts, logistic regression can also be extended to tackle problems involving multiple classes, thereby showcasing its flexibility within supervised learning techniques.

3.6 Gradient Boosting Classifier

Gradient Boosting Classifier constructs a model progressively by successively adding predictors and optimizing differentiable loss functions. At each level, regression trees are fitted to the negative gradients of the binomial or multinomial deviance loss function, depending on the number of classes. This approach combines multiple simple models to produce a powerful predictor noted for its high prediction accuracy.

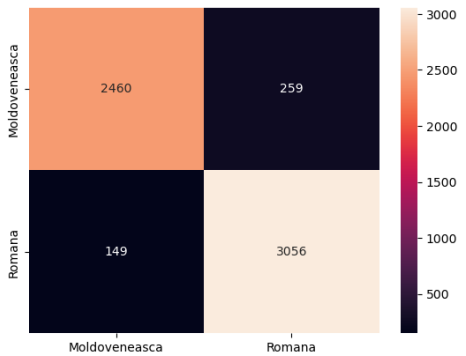

Figure 3: Confusion matrix when using SVC with embeddings computed by RoBERT

4 LLMs for dialect identification

Due to the recent impressive results of LLMs in a diverse range of tasks, we also tried to finetune and evaluate them for the task of dialect identification. We settled on the Mistral language model, as it's

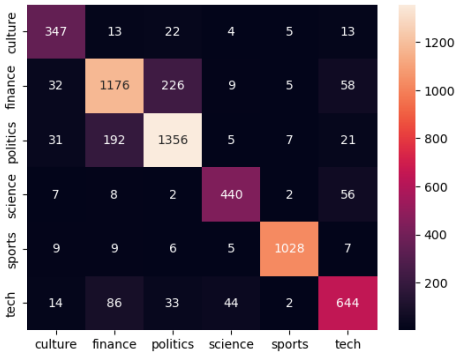

Figure 4: Confusion matrix when using Logistic Regression with embeddings computed by RoBERT

one of the best performing models in its' parameter class (7 billion). Even so, finetuning a LLM is very expensive in terms of compute and memory, and therefore we had to use a number of optimizations to fit in our compute budget.

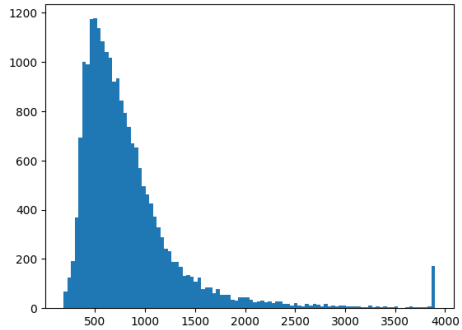

Figure 5: Histogram of context length of MOROCO samples. We clipped sequences with more than 3900 tokens, which consist roughly 0.76% of samples.

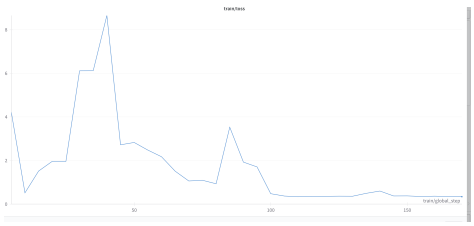

Figure 6: Loss of one of our Mistral training runs.

We used Low Rank Adaptors for training the model, as a full backward pass was unfeasible in our memory regime (24 GB VRAM). We used a rank of 64 and trained only the self-attention weights. For the attention implementation, we used Flash Attention, which enables linear scaling in terms of memory for the sequence length. The model itself was trained using the unsloth library, which provides very efficient fused kernels

for QLoRA finetuning, using 4bit weights for the model. We further used bfloat16 for the computations involving unquantized weights. During training, we used a fused implementation of the AdamW optimizer, using a learning rate of $3e{-}4$, with a cosine learning rate scheduler. Due to problems with convergence, we employed gradient accumulation at 256 samples, and used gradient clipping for gradients with a higher norms than the unit norm. Even so, as it can be seen in one of the loss graphs plotted at Figure 6, the train loss was very noisy.

4.1 Dataset details

Most of the samples in the dataset have under 2000 tokens, with the full distribution of the token count being displayed in Figure 5. Regardless, since we could fit 4096 tokens in memory with our training setup, we decided to use a context length of 4096, in order to include as much as possible of the dataset samples' text, while also having a reasonable training time.

More so, the model we finetuned was based on an instruction tuned version of Mistral. Instruction tuning is a technique used to align LLMs pretrained on vast amounts of tokens to have better responses to human input. It is well known that instruction tuning can greatly increase the performance of LLMs on most tasks, which is why we used the instruction tuned version. In order to respect the instruction tuning format, we had to use the appropriate instruction tokens. To frame the classification task in natural language format, we reframed the task as asking the model to choose between two categories, as in the attached example:

```
[INST] O sa primesti un fragment
↪ dintr-un articol de stiri scris in
↪ limba romana. Trebuie sa il
↪ clasifici in dialectul standard al
↪ limbii romane, sau in dialectul
↪ moldovenesc, folosit in Republica
↪ Moldova. Numele de persoane sau de
↪ locuri geografice au fost schimbate
↪ in "$NE$", ca sa fie impiedicata
↪ folosirea de denumiri specifice
↪ pentru identificare, in loc de
↪ proprietati lingvistice.
Fragmentul este acesta:"<fragment>"
Alege unul dintre cele doua dialecte
↪ pentru clasificare:
1. dialectul moldovenesc
2. dialectul standard
[/INST] Dialectul din fragment este 1.
↪ dialectul moldovenesc</s>
```

4.2 Mistral embeddings

Besides extracting the generated output of the finetuned model, we have also used the model's embeddings for training a suite of classifiers for the dialect identification task. To extract the embeddings, we have computed the average of the embeddings for each token on the last layer, before the classifcation head.

5 Final Results

Given the imbalance in the case numbers for both tasks, we opted for the F1-score as the primary metric to evaluate model performance, as it provides a more comprehensive measure of success than accuracy alone. This choice is particularly advantageous in scenarios where data distribution across classes is uneven, as it balances precision and recall, thus offering a clearer view of a model's effectiveness. Examination of the results, as presented in the tables, reveals that for the task of Dialect Classification, the Support Vector Classifier emerged as the most effective model. Conversely, for the task of categorizing, Logistic Regression proved to be the superior model.

Moreover, it's clear that the effectiveness of classifiers is heavily dependent on the quality of the embeddings used. With embeddings that are more accurate and finely-tuned, produced by more appropriate models, we observe a marked enhancement in the performance across various classifiers. This highlights the critical role that superior-quality embeddings play in boosting the classifiers' precision in making predictions, emphasizing the need for employing advanced and fitting models to generate these embeddings.

It is also perhaps somewhat surprising that Mistral models seem to underperform so much, compared to the more classical methods. One possible reason might be that the base model is very undertrained on Romanian texts, and therefore struggles a lot to generalize on the subtle task of discriminating between the two similar dialects. It's also possible that using a bigger model or training using a less aggressive quantization scheme might lead to better results. Given that the classifiers trained with Mistral embeddings had, in general, significantly better results, it's possible that the sampling process might also negatively influence the final score. Moreover, as the classifiers only deal with the averaged token embeddings for each sample, therefore in some sense having a much lower dimensionality, it's possible that this makes the problem easier to learn with fewer samples.

Classifier	TF-IDF	all-mpnet-base-v2	RoBERT	Mistral
Decision Tree Classifier	0.821	0.645	0.851	0.692
Random Forest Classifier	0.895	0.762	0.921	0.813
Support Vector Classifier	**0.929**	**0.787**	**0.93**	0.565
K-Neighbors Classifier	0.701	0.761	0.93	0.741
Logistic Regression	0.904	0.776	0.929	**0.892**
Gradient Boosting Classifier	0.874	0.733	0.905	0.863

Table 1: In this table, we can see the Accuracy for each classifier on the dialect task

Classifier	TF-IDF	all-mpnet-base-v2	RoBERT
Decision Tree Classifier	0.562	0.515	0.591
Random Forest Classifier	0.711	0.695	0.779
Support Vector Classifier	**0.822**	0.749	0.835
K-Neighbors Classifier	0.492	0.695	0.772
Logistic Regression	0.815	**0.739**	**0.844**
Gradient Boosting Classifier	0.624	0.68	0.782

Table 2: In this table, we can see the F1-Score for each classifier on the categories task

Accuracy	F1
0.55	0.63

Table 3: Results for Mistral generations, the model struggles to generate any correct outputs.

6 Conclusion

We have trained an extensive amount of classifiers on features originating from both classical Machine Learning approaches and deep models. We have also used a Large Language Model for the dialect identification task.

From our results, it seems that, while there isn't a significant difference between the classical and deep features in terms of the greatest performance, the BERT embeddings might have a greater degree of robustness, as they have overall higher separability than the TF-IDF embeddings, when looking at the scores across many different classifiers. Mistral embeddings seem to be somewhat worse than either of them, while also having the highest degree of variance between classifiers.

Comparatively, since Romanian is presumably a low-resource language for the Mistral model's corpus, the performance obtained using it is significantly lower than the other approaches. We hypothesize that using a base model with a much larger Romanian pretraining corpus could lead to a significant improvement.