Projet Natural Language Processing 2025

Language Classification Project

Abstract

Dans le cadre de ce projet de Natural Language Processing (NLP), nous abordons la problématique de l'identification des langues dans un corpus de texte. L'objectif est d'exploiter les connaissances acquises au cours de notre cours pour développer un modèle de **classification** linguistique. Nous explorons différentes approches et méthodes issues du NLP afin d'optimiser la précision de la détection des langues. Ce travail s'inscrit dans une problématique fondamentale du traitement automatique des langues et constitue une application concrète des techniques de classification textuelle.Pour résoudre cette problématique, nous avons choisi d'adopter une approche classique de Machine Learning, avant d'explorer des modèles plus avancés tels que RoBERTa et BERT.

1 Introduction

Dans le cadre de ce projet, notre objectif principal était de développer un modèle capable d'identifier automatiquement la langue de textes fournis dans un corpus. Pour ce faire, nous avons eu accès à deux fichiers `CSV` :

- **test_without_labels.csv** : Un jeu de test contenant des textes à classifier, mais sans leurs labels.
- **train_submission.csv** : Un ensemble de données d'entraînement comprenant les textes et leurs labels, permettant d'entraîner notre modèle.

La tâche à résoudre étant une tâche de **classification**, nous avons structuré notre approche en plusieurs étapes afin d'optimiser la précision et la performance de notre modèle .

2 Steps

- **Exploratory Data Analysis (EDA)** : Analyse préliminaire des données pour comprendre leur distribution et identifier d'éventuels problèmes.Le but est notamment de voir le nombre de langues différentes dans notre corpus de texte,le nombre d'occurrences, les éventuels lignes non-labelisés et de remettre en question la structuration des données . Nous avions alors effectué le nettoyage de notre donnée en créant le fichier: **train_cleaned_numeric_labels.csv'**

- **Tokenization** : La tokenisation consiste à découper les textes en unités linguistiques appelées tokens, ce qui facilite leur traitement par les modèles de machine learning. Nous avons rapidement identifié cette étape comme l'une des principales difficultés du projet, notamment en raison de la diversité des langues utilisées. Nos recherches nous ont orientés vers le modèle de tokenisation *'bert-base-multilingual-cased'*, qui semblait offrir des résultats pertinents. Cependant, étant nous-mêmes polyglottes, nous avons jugé pertinent d'évaluer le comportement de ce tokenizer sur des textes en arabe, une langue généralement plus complexe à décoder.

  À notre grande surprise, nous avons observé une différence notable dans les performances. C'est pourquoi nous avons décidé de nous tourner vers un second tokenizer, *'xlm-roberta-base'*, qui nous a semblé particulièrement adapté et a fourni des résultats bien plus prometteurs. XLM signifie Cross-lingual Language Model. Ce modèle surpasse BERT pour une raison simple : il a été entraîné sur beaucoup plus de données et sur plus de 100 langues, lui donnant un avantage considérable.

  Voici le tableau de comparaison des diverses approches de tokenisation :

Models	Epochs	Optimizer	Batch Size	Learning Rate	Accuracy
BERT	5	Adam	128	1×10^{-5}	60.89%
DistilBERT	5	Adam	128	1×10^{-5}	58.09%
XLM-RoBERTa	5	Adam	128	1×10^{-5}	67.66%

Table 1: Performance des modèles de tokenization

- **Models** : Expérimentation avec différentes approches, allant des modèles classiques de Machine Learning aux modèles avancés comme `BERT` et `RoBERTa`.

L'ensemble du code et des expérimentations associées est disponible sur notre dépôt GitHub : GitHub Project Repository.

3 Solutions

Nous avons ensuite décidé d'appliquer les modèles identifiés lors de notre recherche. Ces modèles ont été exécutés sur nos ordinateurs personnels pendant plusieurs heures, voire une journée entière, en raison de l'absence d'accès à une machine plus performante équipée d'un GPU capable d'accélérer les calculs. Cette limitation matérielle a constitué notre principale difficulté, ralentissant considérablement notre progression.

Actuellement, notre code avec lee meilleur résultat a atteint une accuracy de 0.73 et comprend les étapes suivantes :

- **Encodage des labels**
- **Nettoyage des textes**, avec une limitation à **256 tokens**
- **Séparation Train / Validation**
- **Utilisation du modèle RoBERTa** pour la classification
- **Tokenisation des textes**
- **Hyperparamètres du modèle** :
  - `learning_rate` = **2e-5**
  - `epochs` = **5**
  - `optimizer` = **AdamW** avec `weight_decay=0.01`
  - `scheduler` = `get_scheduler("linear", optimizer=optimizer, num_warmup_steps=1000, num_training_steps=epochs * len(train_loader))`
- **Entraînement du modèle**
- **Évaluation sur l'ensemble de validation**

Au terme de ces expérimentations, nous avons constaté que :

- DistilBERT offre un bon compromis entre rapidité et précision.
- XLM-RoBERTa assure les meilleures performances pour le traitement multilingue.
- BERT Multilingual constitue un modèle stable et standard. Pour mener ces tests, nous avons utilisé le code développé précédemment.

4 Ameliorations

Suite à nos expérimentations, plusieurs pistes d'améliorations ont été identifiées pour augmenter la performance et la précision du modèle. Parmi les principales améliorations que nous envisageons, on peut citer :

- **Utilisation d'un GPU** : L'un des principaux obstacles à l'amélioration de nos résultats a été le manque de ressources matérielles. L'utilisation d'un GPU pour accélérer l'entraînement permettrait de tester des modèles plus complexes et de réduire significativement le temps d'entraînement.
- **Ensembles de modèles** : Tester des approches d'ensembles de modèles, comme les approches de vote ou de bagging, pourrait renforcer la précision du modèle final en combinant plusieurs prédictions de différents modèles.
- **Fine-tuning des modèles pré-entrainés** : Bien que l'on ait utilisé des modèles comme RoBERTa et XLM-RoBERTa, un fine-tuning plus approfondi, incluant l'adaptation à notre jeu de données spécifique, pourrait offrir de meilleurs résultats.

5 Conclusion

En conclusion, ce projet a permis d'identifier plusieurs solutions pour la tâche de classification linguistique, notamment l'utilisation de modèles avancés comme RoBERTa et XLM-RoBERTa. Malgré les limitations matérielles qui ont ralenti notre avancement, nous avons réussi à obtenir des résultats encourageants, avec une précision de **0.73**. Ces résultats confirment que les modèles de traitement du langage naturel tels que RoBERTa peuvent être très efficaces pour la classification des langues, même dans un contexte multilingue.

Les améliorations futures pourraient inclure une meilleure exploration des hyperparamètres, l'utilisation de ressources plus puissantes pour l'entraînement, et des techniques d'augmentation de données pour rendre le modèle encore plus robuste. Ce travail a permis de poser les bases solides pour des recherches futures et d'identifier des pistes d'amélioration pour la classification linguistique à grande échelle.

References

[1] Vtiya, The Difference of BERT, XLM-RoBERTa, and Longformer, *Medium*, 2021, https://vtiya.medium.com/the-difference-of-bert-xlm-roberta-and-longformer-c23aee338297.

[2] DataScientest, Introduction au NLP (Natural Language Processing), *DataScientest*, 2021, https://datascientest.com/introduction-au-nlp-natural-language-processing.

[3] Hugging Face, Tokenizer Summary, *Hugging Face Documentation*, 2021, https://huggingface.co/docs/transformers/tokenizer_summary.

[4] Hugging Face, jmedroberta-base-sentencepiece, *Hugging Face Model*, 2021, https://huggingface.co/alabnii/jmedroberta-base-sentencepiece.