# ANLP Kaggle project
## Language identification (LID)

Ilann Amiaud–Plachy[1], Thomas Bodart[1], Maceo Duriez[1], Marc-César Garcia-Grenet[1], Gaétan Jacquemin[1]

[1]CentraleSupélec, Paris-Saclay University

**Abstract**

*L'identification de langue (LID) est une tâche essentielle en traitement du langage naturel (NLP), jouant un rôle clé dans de nombreuses applications multilingues. Dans le cadre du projet Kaggle du cours d'ANLP, nous avons exploré différentes approches, à la fois statistiques et basées sur l'apprentissage profond, afin de développer un modèle performant de LID. Après comparaison, notre solution finale combine FastText, rapide et efficace pour la classification de texte, avec roBERTa, un modèle transformer offrant une meilleure capacité de généralisation, en utilisant les probabilités données par les modèles.*

## Introduction

La détection automatique de la langue représente un enjeux pour le traitement automatique du language, on s'en sert pour la traduction et pour l'organisation automatique de contenus multilingues, très utile pour la collecte de données à grand échelle, permettant de rassembler et labeliser des corpus de données de manière automatique

Les approchent traditionnelles de détection linguistique sont surtout des approches statistiques comme les n-grammes [1] ou TF-IDF [2]. Mais aujourd'hui il est pertinent d'évaluer la performance des modèles de language neuronaux pré-entrainés sur des grands corpus multilingues comme BERT [3].

Dans notre étude, nous avons exploré et comparé plusieurs approches :

- Une approche classique utilisant TF-IDF avec un classifieur SVM
- FastText, qui exploite efficacement les représentations de sous-mots [4]
- Des modèles transformers multilingues comme mBERT [3] et XLM-RoBERTa [5]

La suite de ce rapport est organisée comme suit : la section 2 présente plus précisémenent les solutions que nous avons essayés, et la section 3 présente et discute des résultats.

## Solutions

### Exploration du dataset

Le fichier d'entraînement train_submissions.csv proposé pour cette compétition kaggle est un fichier csv de 3 colonnes Usage, Text, Label et 190599 lignes avec 389 labels cibles. Parmi ces 190599 lignes, 500 d'entre elles ne possèdent pas de label.

On peut noter que les labels sont déséquilibrés, comme le montre la figure suivante :

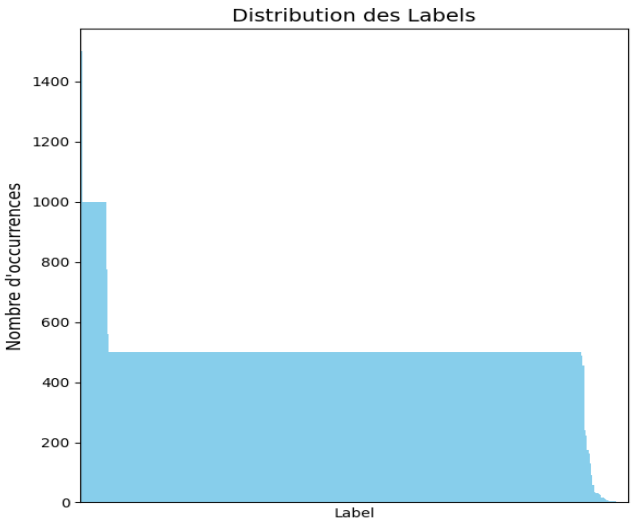


**Figure 1:** *Distribution déséquilibrée des labels dans le dataset*

Nous avons essayé plusieurs techniques de downsampling et d'upsampling pour palier au déséquilibre des données, notamment en essayant d'enrichir les labels avec le moins de données disponibles grâce à des LLMs. Néanmoins cette approche n'a pas donné de bon résultats car cela retirait une partie des biais inhérents au dataset. Nous avons donc fait le choix de downsampler les labels ayant plus de 500 labels en choisissant 500 lignes aléatoirement parmi ces labels.

Il est à noter que le dataset comprenait de nombreuses langues régionales très peu parlées donc mal prises en compte par de nombreux modèles pré-entrainé. De plus, pour ces langues, la quantité de données disponibles était assez faible puisque plus de 50% des textes ne comportaient pas plus de 16 mots.

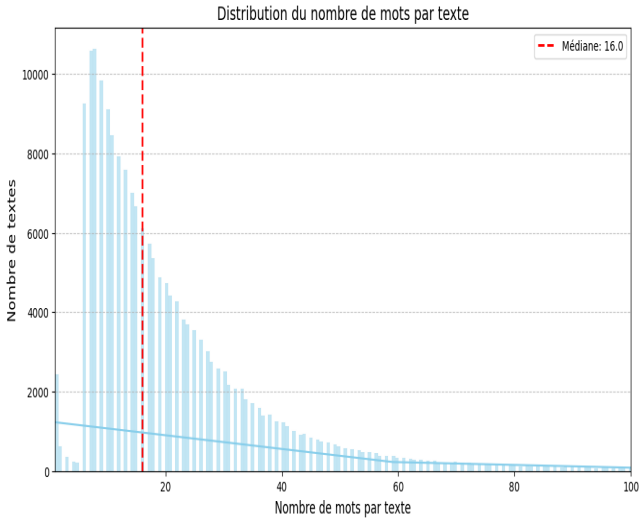


**Figure 2:** *Nombre de mots par texte.*

## Preprocessing

Les données sont parfois bruitées : certaines contiennent du code html, des adresses mails ou encore des emojis. En outre, certaines données contiennent beaucoup de mots en anglais (ou autre langue très parlée), voire sont mal classifiées : cela s'explique notamment par le fait que les données proviennent de site tels que Twitter/X, où beaucoup d'utilisateurs utilisent l'anglais par soucis de compréhension. Afin d'éliminer ce bruit, nous avons utilisés deux preprocessings: suppression des charactères non pertinents (emojis, balises html, ...) et suppression des mots en anglais, français et espagnol lorsqu'ils sont minoritaires dans la phrase.

Pour la suppression des caractères non pertinents, nous avons utilisé des règles regex. L'impact sur la qualité des prédictions n'a pas été très important : cela pourrait s'expliquer par la suppression de biais présents dans certaines langues, biais qui proviennent du scrapping de sources similaires et qui créent des similarités intra-classes ne provenant pas du langage. Par exemple, on pourrait imaginer que les données disponibles pour une langue ont été intégralement prises d'un compte Twitter/X, ce qui pourrait créer un biais sur l'utilisation de certains emojis.

Pour la suppression des mots en anglais, français, espagnol, nous avons utilisé des bases de données de mots provenant d'un github ([6]). Nous avons utilisé un seuil : si les mots repérés sont minoritaires dans la phrase (moins de 50% de la phrase), nous avons considéré que ces mots étaient empruntés à l'anglais, au français ou à l'espagnol, et avons retiré ces mots des phrases. Cette démarche n'a pas été très concluante : cela s'explique notamment par la présence de dialectes (par exemple, l'argentin) qui partagent de nombreux mots avec d'autres langues, et de laisser les biais du dataset être appris par nos modèles.

Finalement, nous avons choisi d'utiliser le dataset tel quel, et de concentrer nos efforts sur les modèles utilisés.

## TF-IDF

TF-IDF est une méthode statistique utilisée pour évaluer l'importance d'un mot dans un document par rapport à une collection de documents. Elle combine deux métriques : la fréquence du terme (TF) et la fréquence inverse du document (IDF). La fréquence du terme mesure combien de fois un mot apparaît dans un document, tandis que l'IDF diminue l'importance des mots qui apparaissent fréquemment dans tous les documents.

Pour notre utilisation, TF-IDF nous permet de créer des vecteurs de représentation à partir des textes, puis d'entraîner un classifieur pour évaluer la langue du texte. Dans notre cas, nous avons utilisé un classifieur SVM car c'est celui qui, théoriquement et expérimentalement, donne les meilleurs résultats (voir [2]).

Cette approche nous a donné des résultats assez satisfaisants mais pas suffisants pour notre cas d'utilisation et s'est laissé largement dépasser par les autres approches, certainement car elle ne prend pas en compte les relations entre les mots et les contextes dans lesquels ils apparaissent.

## FastText

FastText est une bibliothèque développée par Facebook [4] pour la représentation de mots et l'apprentissage de classificateurs de texte. Comme vu en cours, elle repose sur des modèles de type n-grammes pour capturer des informations sur les sous-mots, ce qui permet de mieux gérer les mots rares et les fautes d'orthographe.

FastText est particulièrement efficace pour l'identification de langue car il peut capturer des caractéristiques spécifiques à chaque langue au niveau des sous-mots. Il a donné de très bons résultats dans notre étude, avec des scores similaires voire supérieurs aux modèles de langage neuronaux (voir ci-dessous), mais avec des temps d'entraînement et d'inférence presque instantanés en comparaison.

Pour son entraînement, nous avons dans un premier temps pris les hyperparamètres donnés dans l'article GlotLID [7], un article présentant un modèle classifiant plus de 1800 langues, puis nous avons fait une recherche d'hyperparamètres aux alentours de ces derniers.

| Argument | Description | Valeur |
|---|---|---|
| -minCount | Occurrences min mot | 1000 |
| -minCountLabel | Occurrences min label | 0 |
| -wordNgrams | Max n-grammes mots | 1 |
| -bucket | Nb de buckets | $10^6$ |
| -minn | Min. n-gram caractères | 1 |
| -maxn | Max. n-gram caractères | 5 |
| -loss | Perte | softmax |
| -dim | Taille vecteurs | 256 |
| -epoch | Nombre d'époques | 9 |
| -lr | Taux apprentissage | 1.5 |

**Table 1:** *Hyperparamètres d'entraînement du modèle FastText*

### mBERT

mBERT (Multilingual BERT) est la version multilingue du modèle BERT [3], qui est un modèle de langage basé sur des transformers, possédant 179M de paramètres. Sa particularité est qu'il est pré-entraîné sur la tâche de Masked Language Model, sur un vaste corpus de textes dans 104 langues, ce qui lui permet de comprendre et de générer des représentations contextuelles pour des textes dans différentes langues.

Ce modèle fut notre première approche de langage neuronal pour ce projet. Nous avons de même utilisé le tokenizer correspondant à mBERT, comprenant tous les caractères de notre corpus, et nous n'avons pas mis de limite de taille pour le nombre de tokens. Nous avons d'abord essayé une méthode de transfert (transfer learning) pour adapter mBERT à notre tâche, mais cela s'est avéré insuffisant et il a finalement fallu ajuster les poids du modèle (fine-tuning) pour obtenir des résultats satisfaisants sur 10 epochs, en utilisant 85% du dataset fourni en train, et une dernière epoch en utilisant la totalité du corpus, pour utiliser le plein potentiel de nos données. L'inconvénient de cette méthode est qu'elle est très lourde en calculs (plusieurs heures d'entraînement).

Finalement, mBERT a donné des résultats satisfaisants mais s'est logiquement laissé dépasser par XLM-RoBERTa (voir ci-dessous), qui possède en particulier plus de paramètres.

### XLM-RoBERTa

XLM-RoBERTa [5] est une variante optimisée de BERT [3] qui utilise un processus d'entraînement plus robuste et une plus grande quantité de données sur 100 langues. Nous avons utilisé le modèle base à 279M de paramètres. Les améliorations incluent l'entraînement sur des lots de données plus grands, l'utilisation de plus de données d'entraînement, et l'optimisation des hyperparamètres. C'est donc très logiquement que nos résultats suivent l'amélioration promise par les chercheurs à son origine et qu'il surpasse mBERT.

Notre utilisation reprend la même logique que pour mBERT avec une nette amélioration des performances, mais avec un coût en temps d'entraînement et d'inférence plus élevé. Pour palier à cela, nous avons décidé de restreindre chaque phrase à 128 tokens, avec un padding si la phrase contient moins de 128 tokens, et un troncage si elle en contient plus. Cela a drastiquement réduit le temps d'entraînement. Nous avons également utilisé un taux d'apprentissage de 2e-5, un batch size de 64, et un nombre d'époques de 10. On a de plus utilisé le tokenizer de XLM-RoBERTa.

### Combinaison FastText et XLM-RoBERTa

Il nous est naturellement venu l'envie de combiner nos modèles portant sur des approches différentes pour tenter de profiter des avantages de chaques approches, pour améliorer les performances de notre solution. La combinaison qui nous a donné les meilleurs résultats après divers essais est la suivante : si l'inférence avec FastText donne une prédiction avec une probabilité supérieure à 0.99%, on la garde ; sinon, on utilise XLM-RoBERTa pour donner une prédiction. Nous avons fait une étude de sensibilité sur le seuil de probabilité de FastText, et 0.99% est le seuil qui nous a donné les meilleurs résultats, et, en effet, les prédictions au dessus de ce seuil ont une accuracy de 99% sur le val.

## Resultats et Analyses

Pour évaluer la performance de nos modèles, nous utilisons l'accuracy comme métrique principale. Cette métrique est particulièrement pertinente dans notre cas, car les classes sont relativement équilibrées dans notre corpus, y compris dans l'ensemble de test.

Les résultats obtenus pour les différents modèles sont présentés dans le tableau 2.

| Modèle | Val acc | Test acc | inférence* |
|---|---|---|---|
| FastText | 0.86 | 0.84 | 1min |
| TFIDF | 0.34 | 0.33 | 30s |
| mBERT | 0.86 | 0.86 | 1h15 |
| XLM-roBERTa | 0.87 | 0.87 | 1h |
| FT + XLM-R | 0.88 | 0.88 | 1h |

**Table 2:** *Accuracy sur le val et sur le test des modèles essayés. (*durée d'inférence sur un ordinateur classique avec GPU sur tout le corpus de texte)*

### Performance globale des modèles

Comme attendu, les modèles neuronaux de grande taille (mBERT, XLM-RoBERTa) offrent des performances élevées, avec une accuracy test atteignant 0.86 et 0.87 respectivement. Ces résultats confirment que les modèles de type Transformer sont particulièrement efficaces pour la tâche de détection de langues, grâce à leur capacité à capturer des représentations riches et contextuelles des textes.

Toutefois, cette performance a un coût : le temps d'inférence est particulièrement élevé. Par exemple, mBERT nécessite **1h15** pour traiter l'ensemble du corpus, ce qui peut poser des contraintes importantes dans un contexte de production ou de traitement en temps réel.

Le modèle TF-IDF, basé sur une approche purement statistique, affiche des résultats bien inférieurs (**0.34 d'accuracy**), ce qui montre ses limites face aux approches basées sur l'apprentissage automatique. Cela est sûrement dû à la présence de dialectes très proches.

### L'exception FastText

Un résultat particulièrement intéressant est celui de **FastText**, qui atteint **0.84 d'accuracy en test**, soit un score très proche de ceux obtenus avec des modèles neuronaux beaucoup plus lourds. Ce modèle présente un énorme avantage : son **temps d'inférence est quasi instantané** (1 min sur l'ensemble du corpus de 200K instances). Cela en fait un candidat idéal pour des applications nécessitant un traitement rapide et efficace, comme la classification de textes en temps réel ou l'analyse de grandes quantités de données avec des ressources limitées.

### Vers une combinaison optimale : FT + XLM-R

Face à ces observations, nous avons cherché à combiner les forces de nos différents modèles pour accroître l'accuracy et être plus performant sur le concours. L'association de **FastText et XLM-RoBERTa** (FT + XLM-R) a permis d'atteindre une **accuracy de 0.88**, soit la meilleure performance obtenue parmi tous les modèles testés. En combinant ces deux modèles, nous obtenons un système performant qui optimise le **l'accuracy** bien qu'elle ne permette pas de réduire le temps d'inférence.

## Conclusion et perspectives

Nos résultats montrent que, bien que les modèles de type Transformer offrent d'excellentes performances en détection de langues, leur coût en calcul reste un défi majeur. FastText s'est révélé être une alternative étonnamment efficace, capable d'obtenir des résultats proches des modèles neuronaux tout en étant extrêmement rapide.

La combinaison de FastText avec XLM-RoBERTa permet de tirer parti des avantages des deux approches, offrant ainsi une solution optimisée pour la détection de langues à grande échelle. Dans le futur, des recherches pourraient être menées pour affiner davantage le temps d'inférence de notre meilleur modèle tout en conservant son excellente précision, notamment en explorant des techniques comme la **distillation de connaissances**, où un modèle léger apprend à reproduire les décisions d'un modèle plus complexe. Pour aller plus loin au niveau de la précision, il serait intéressant d'explorer plus en profondeur, et avec plus de temps les hyperparamètres de nos grands modèles de langages neuronaux, pour voir si nous pouvons encore améliorer les performances, voir de tester, avec une puissance de calcul plus importante, des modèles plus grands comme RoBERTa large, possèdant 355M de paramètres.

## References


[1] William B Cavnar and John M Trenkle. "N-gram-based text categorization". In: *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*. 1994, pp. 161–175.

[2] Timothy Baldwin and Marco Lui. "Language identification: The long and the short of the matter". In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL* (2010), pp. 229–237.

[3] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of NAACL-HLT 2019* (2019), pp. 4171–4186.

[4] Armand Joulin et al. "Bag of Tricks for Efficient Text Classification". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics* (2017), pp. 427–431.

[5] Alexis Conneau et al. "Unsupervised Cross-lingual Representation Learning at Scale". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), pp. 8440–8451.

[6] Hermit Dave. *Frequency Words - 2018*. Accessed: 2023-10-15. 2018. URL: `https://github.com/hermitdave/FrequencyWords/tree/master`.

[7] Amir Kargaran et al. "GlotLID: Language Identification for Low-Resource Languages". In: *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, 2023, pp. 6155–6218. DOI: 10.18653/v1/2023.findings-emnlp.410. URL: http://dx.doi.org/10.18653/v1/2023.findings-emnlp.410.