



UNIVERSITÉ
CÔTE D'AZUR

UNIVERSITÉ CÔTE D'AZUR

MASTER INFORMATIQUE
TRAITEMENT AUTOMATIQUE DU TEXTE EN IA
RAPPORT

Language Detector

Students :
MOHAMED Belhassen

Instructor :
Cabrio ELENA

December 28, 2021

Contents

1	Introduction et présentation du sujet	2
1.1	Présentation du sujet	2
1.2	Jeu de données	2
2	Conception et implémentation	3
2.1	Extraction de données	3
2.2	Preprocessing	3
2.3	Vectorisation et Classification	4
2.3.1	TF-IDF vs Count Vectorizer	4
2.3.2	Analyzer : Char vs Word	4
2.3.3	N_gram : 1-gram 2-gram ou 3-gram	5
2.3.4	all options together	6
3	Résultats obtenus et problèmes rencontrés	7
3.1	Résultats obtenus	7
4	Conclusion	7
5	Améliorations possibles	7
6	Link Github	8

1 Introduction et présentation du sujet

1.1 Présentation du sujet

Nous avons choisi de créer un modèle qui nous permet Détecter la langue d'un texte donné .nous sommes inspiré par les éditeurs des textes et les outils de translation en ligne (DeepL, google Translate, Reverso) qui ils implémentent cette fonctionnalité . Nous avons choisi des langues qui ils sont dérivés de même famille latine parce que : premièrement ils utilisent les mêmes alphabètes pour mieux tester l'efficacité de notre modèle d'apprentissage sinon ça sera facile de detecter des langues des différentes familles (anglais, Arabes, chinois) .

Pour le moment , notre détecteur supporte 5 langues :

- Anglais
- Français
- Italien
- Espagnol
- Allemand

1.2 Jeu de données

En ce qui concerne le jeu de données, nous avons utilisé the European Parliament Proceedings Parallel Corpus comme source. les données sont extraites des actes du Parlement européen. le corpus comprend des versions dans 21 langues européennes.

Les données peuvent être téléchargées ici .






Nom	Modifié le	Type	Taille
 english.txt	27/12/2021 16:17	Document texte	15 071 Ko
 french.txt	27/12/2021 16:20	Document texte	16 798 Ko
 german.txt	27/12/2021 16:20	Document texte	16 694 Ko
 italian.txt	27/12/2021 16:20	Document texte	16 908 Ko
 spanish.txt	27/12/2021 16:20	Document texte	16 218 Ko

Figure 1: Data files

2 Conception et implémentation

2.1 Extraction de données

Pour l'extraction des données, nous avons importé nos fichiers textes et nous avons créé un data frame pour chaque langue. Environ 8 000 lignes de texte ont été sélectionnées au hasard pour chaque langue. Ensuite, une fois que nous avons créé nos DATA FRAMES avec les données textes et ajouter une colonne indique la langue de ces derniers, il fallait les concaténer dans une seule data frame.

	sentence	language
0	J'espère que la Commission peut accepter cet a...	french
1	Signora Presidente, volevo dire all' onorevole...	italian
2	Au contraire, nous voulons une Union européen...	french
3	Es gibt jedoch noch viel zu tun, und ich forde...	german
4	El presidente del Partido Liberal austríaco, H...	spanish
5	J'en viens maintenant brièvement au rapport de...	french
6	It is, however, the best that we could achieve...	english
7	Es evidente que, desde el punto de vista insti...	spanish
8	Inoltre, negli anni sin qui trascorsi l'accent...	italian
9	I have also offered the Spanish authorities ou...	english

Figure 2: Dataframe

2.2 Preprocessing

Pour la suite, nous avons créé une fonction removeNoise pour le prétraitement des données. Cette fonction permet de :

- mettre tout le texte en minuscule
- supprimer les ponctuations et les symboles.
- supprimer les chiffres.

	sentence	language
0	j espère que la commission peut accepter cet a...	french
1	signora presidente volevo dire all onorevole...	italian
2	au contraire nous voulons une union européen...	french
3	es gibt jedoch noch viel zu tun und ich forde...	german
4	el presidente del partido liberal austríaco h...	spanish
5	j en viens maintenant brièvement au rapport de...	french
6	it is however the best that we could achieve...	english
7	es evidente que desde el punto de vista insti...	spanish
8	inoltre negli anni sin qui trascorsi l accent...	italian
9	i have also offered the spanish authorities ou...	english

Figure 3: Dataframe without noise

Dans le prétraitement, nous séparons aussi nos données de « data » en deux afin d'entraîner et tester notre machine supervisée (80% d'entraînement et 20% de test de prédiction).

2.3 Vectorisation et Classification

Afin de trouver le meilleur modèle et les meilleurs paramètres, nous allons tester plusieurs composant et selon l'évaluation de notre modèle, on va garder les paramètres optimisés

2.3.1 TF-IDF vs Count Vectorizer

On voulait initialement comparer les deux transformateurs **Count Vectorizers** et **TF-IDF**. nous les testons avec l'estimateur **régression logistique** qui est largement utilisé pour traiter les problèmes de classification.

donc nous avons obtenu un excellent accuracy score pour les deux modeles avec une légère différence pour **TF-IDF** 99,6% contre 99,5% pour **CountVectorizer**

2.3.2 Analyzer : Char vs Word

Ensuite nous allons comparer les deux types d'**Analyzer Char** et **word**. Nous les testons avec l'estimateur **régression logistique** et le **TF-IDF Vectorizer**. Nous utilisons un pipeline pour mettre en œuvre ces modèles.

	params	rank_test_score	mean_test_score	std_test_score
Analyzer				
word	{'tfidfvectorizer__analyzer': 'word'}	1	0.996125	0.001043
char	{'tfidfvectorizer__analyzer': 'char'}	2	0.969031	0.002161

Figure 4: word vs char

donc comme nous le voyons, nous avons à nouveau un grand score de précision pour le **char** et le **word** qui est plus élevé de 3%. ce qui m'a surpris car sachant que toutes les langues ici **ont en commun presque le même alphabet et la principale différence est la façon dont ils apparaissent dans le mot**, je pensais que le char aurait une meilleure précision.

2.3.3 N_gram : 1-gram 2-gram ou 3-gram

Nous allons comparer la modélisation des ngrams (de **1-gram** à **3-gram**). nous les testons avec l'estimateur **régression logistique** et le **TF-IDF Vectorizer** et **word Analyzer**. Nous utilisons un pipeline pour mettre en œuvre ces modèles.

	params	rank_test_score	mean_test_score	std_test_score
Analyzer				
(1, 2)	['tfidfvectorizer__ngram_range': (1, 2)]	1	0.995875	0.000660
(1, 1)	['tfidfvectorizer__ngram_range': (1, 1)]	2	0.995813	0.000673
(1, 3)	['tfidfvectorizer__ngram_range': (1, 3)]	3	0.995781	0.000765

Figure 5: n-gram-range

- Une précision de prédiction de 99,5875 % a été obtenue sur les données de test à l'aide d'un modèle formé avec le modèle de régression logistique du mot à 2-gram
- Une précision de prédiction de 99,5813 % a été obtenue sur les données de test à l'aide d'un modèle entraîné avec un modèle de régression logistique de mots de 1-gram.
- Une précision de prédiction de 99,5781 % a été obtenue sur les données de test à l'aide d'un modèle entraîné avec un modèle de régression logistique de mots de 3-gram.

2.3.4 all options together

dans cette dernière partie, nous allons tester toutes les options de Vectorizer ensemble :

- 1-gram Character frequency analysis
- 2-gram Character frequency analysis
- 3-gram Character frequency analysis
- 1-gram Word frequency analysis
- 2-gram Word frequency analysis
- 3-gram Word frequency analysis

	params	rank_test_score	mean_test_score	std_test_score
Analyzer\n_gram				
char_(1, 3)	{'tfidfvectorizer__analyzer': 'char', 'tfidfve...	1	0.996438	0.000468
word_(1, 2)	{'tfidfvectorizer__analyzer': 'word', 'tfidfve...	2	0.995875	0.000660
word_(1, 1)	{'tfidfvectorizer__analyzer': 'word', 'tfidfve...	3	0.995813	0.000673
word_(1, 3)	{'tfidfvectorizer__analyzer': 'word', 'tfidfve...	4	0.995781	0.000765
char_(1, 2)	{'tfidfvectorizer__analyzer': 'char', 'tfidfve...	5	0.994563	0.000636
char_(1, 1)	{'tfidfvectorizer__analyzer': 'char', 'tfidfve...	6	0.967656	0.001269

Figure 6: all options together

Nous avons constaté que notre modèle est encore plus performant avec analyzer = char et n_gram = (1,3) Dans la prochaine sections, nous verrons toutes les métriques et nous essaierons de les analyser.

3 Résultats obtenus et problèmes rencontrés

3.1 Résultats obtenus

Nous allons maintenant voir et comparer les résultats que nous avons obtenus. Après avoir entraîné nos différents modèles sur les 80% dédiés à l'entraînement pur, nous faisons des prédictions sur les 20% restants :

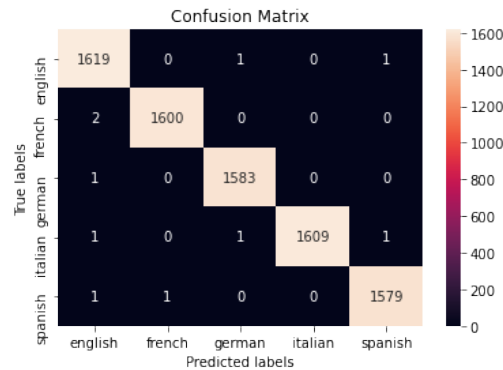


Figure 7: all options together

like we see ,our model performance is greate , we have only :

- - 2 french strings were missclassified as english
- - 1 german strings were missclassified as english
- - 1 italien strings were missclassified as english
- - 1 spanish strings were missclassified as english
- - 1 english strings were missclassified as german
- - 1 english strings were missclassified as spanish

4 Conclusion

pour conclure, nous avons vu que la performance de notre modèle dépend de plusieurs variables, sachant que nous obtenons toujours un score élevé qui dépasse le 95% .

Cela nous ramène donc aux Améliorations possibles dans notre projet

5 Améliorations possibles

Nous pourrions fpar exemple ajouter des autres langues.Aussi pour améliorer notre modele , Nous pouvons tester d'autres modèles (Le modèle **Linear SVC** ou Le modèle **Decision Tree**). Nous pouvons également essayer des modèles plus élevés de type n-gram, mais les coûts de calcul et le temps requis pour former ces modèles peuvent être très élevés pour un PC ordinaire.

En ce qui concerne les autres applications possibles , on integrer l'analyse de sons mais dans ce cas on sort du cadre du cours.

6 Link Github

Github : <https://github.com/mohamedlouay/LanguageDetector> .