DataScientest

DÉMOSTHÈNE

Projet de traduction
pour lunettes connectées

> *Je ne crois pas qu'on trouve ailleurs un orateur qui se soit élevé, comme lui, du sein de l'obscurité et de la faiblesse, à ce haut degré de puissance et de gloire.*

Plutarque, « Vies des hommes illustres »

RÉSUMÉ

Les lunettes connectées permettent d'afficher la transcription de la voix d'un interlocuteur. Cette transcription peut être traduite dans la langue du porteur de ces lunettes. Le projet s'est focalisé sur le système de traduction

Équipe du projet
Keyne DUPONT
Tia RATSIMBASON
Olivier RENOUARD

Mentor
Thomas BOEHLER

Chef de cohorte
Manon GEORGET

À propos
Ce projet a été réalisé dans le cadre d'une formation de Data Scientist, entre juin et novembre 2023.

Démosthène est l'un des plus grands orateurs de l'Antiquité. il savait se faire s'exprimer, et se faire comprendre. Se faire comprendre est l'un des principaux objectifs de la traduction.

Démosthène avait de gros problèmes d'élocution. Il les a surmontés en s'entraînant à parler avec des cailloux dans la bouche, à l'image de l'Intelligence Artificielle, où des entraînements sont nécessaires pour obtenir de bons résultats. Il nous a semblé pertinent de donner le nom de cet homme à un projet qu'il a fort bien illustré, il y a 2300 ans.

À l'attention du lecteur
Nous avons souhaité offrir un ordre de lecture logique, mais qui ne suit pas celui du travail effectué. Chronologiquement, nous avons travaillé sur les parties 2, 3, 5, 6 pour les RNN et Transformers Keras, puis 4. Enfin, nous avons réabordé la partie 6 pour les Transformers Hugging Face et Text2Speech.

Les ressources produites sont consultables dans les notebooks, et l'application Streamlit, et le projet Github .

Table des matières

1.Introduction

a. Contexte

Les personnes malentendantes communiquent difficilement avec autrui. Par ailleurs, toute personne se trouvant dans un pays étranger dont il ne connaît pas la langue se retrouve dans la situation d'une personne malentendante.

L'usage de lunettes connectées, dotées de la technologie de reconnaissance vocale et d'algorithmes IA de deep learning, permettrait de détecter la voix d'un interlocuteur, puis d'afficher la transcription textuelle, sur les verres, en temps réel.

À partir de cette transcription, il est possible d'afficher la traduction dans la langue du porteur de ces lunettes.

b. Cadrage de la problématique

L'objectif de ce projet est de développer une brique technologique de traitement, de transcription et de traduction, qui par la suite serait implémentable dans des lunettes connectées.

Pour ce projet, nous avons concentré nos efforts sur la construction d'un système de traduction plutôt que sur la reconnaissance vocale, et ce, pour tout type de public, afin de faciliter le dialogue entre deux individus ne pratiquant pas la même langue.

Il est bien sûr souhaitable que le système puisse rapidement identifier la langue des phrases fournies, à condition qu'elle soit connue.

Lors de la traduction, nous ne prendrons pas en compte le contexte des phrases précédentes ou celles préalablement traduites.

Nous pourrons évaluer la qualité de nos résultats en les comparant avec des systèmes performants tels que "Google translate" et "Deepl".

2. Exploration des données et pré-processing

a. Jeux de données utilisés

1. Small_vocab

Proposé par Suzan Li, Chief Data Scientist chez Campaign Research à Toronto, ce dataset représente un corpus de phrases simples en anglais et sa traduction, approximative, en français.

137 860 phrases en & fr
1 552 863 mots anglais dont 199 mots uniques
1 728 899 mots français dont 330 mots uniques.

Référence
- *Gitbub "NLP with Python" Suzan Li*
- *Données du projets*

2. Vectors-Wiki

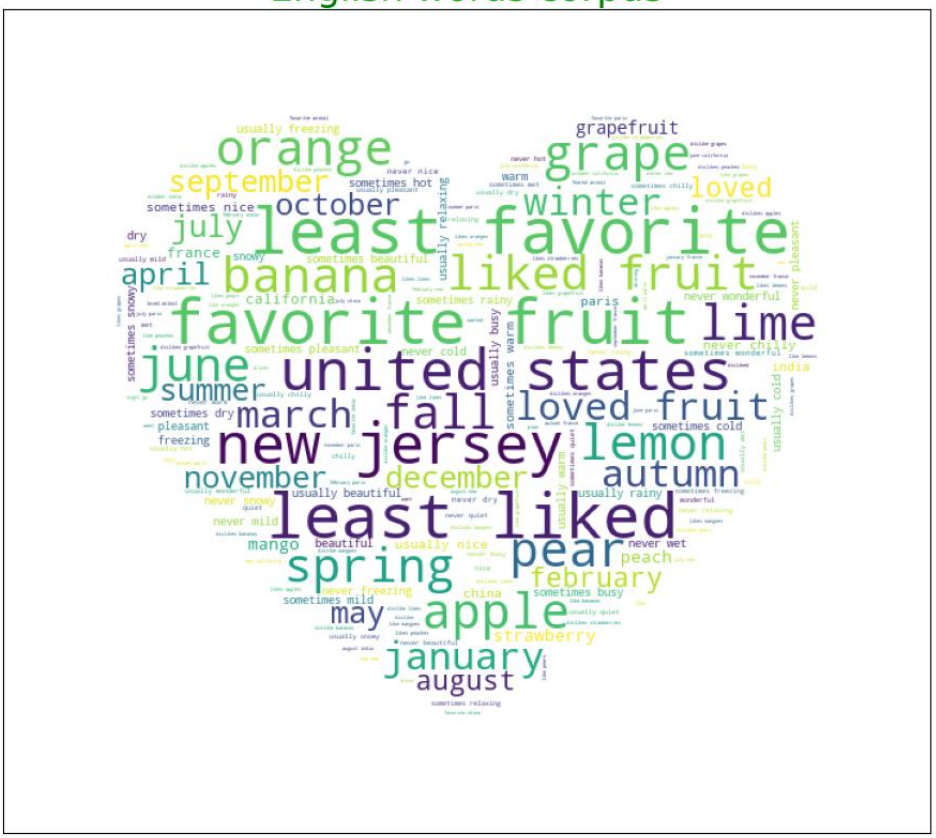
Plusieurs millions de mots monolingues, vectorisés dans des espaces vectoriels alignés, issus de corpus parallèles de Wikimedia compilés par Facebook Research.

2 519 370 mots anglais uniques
1 152 449 mots français uniques avec ses vecteurs de 300 dimensions.

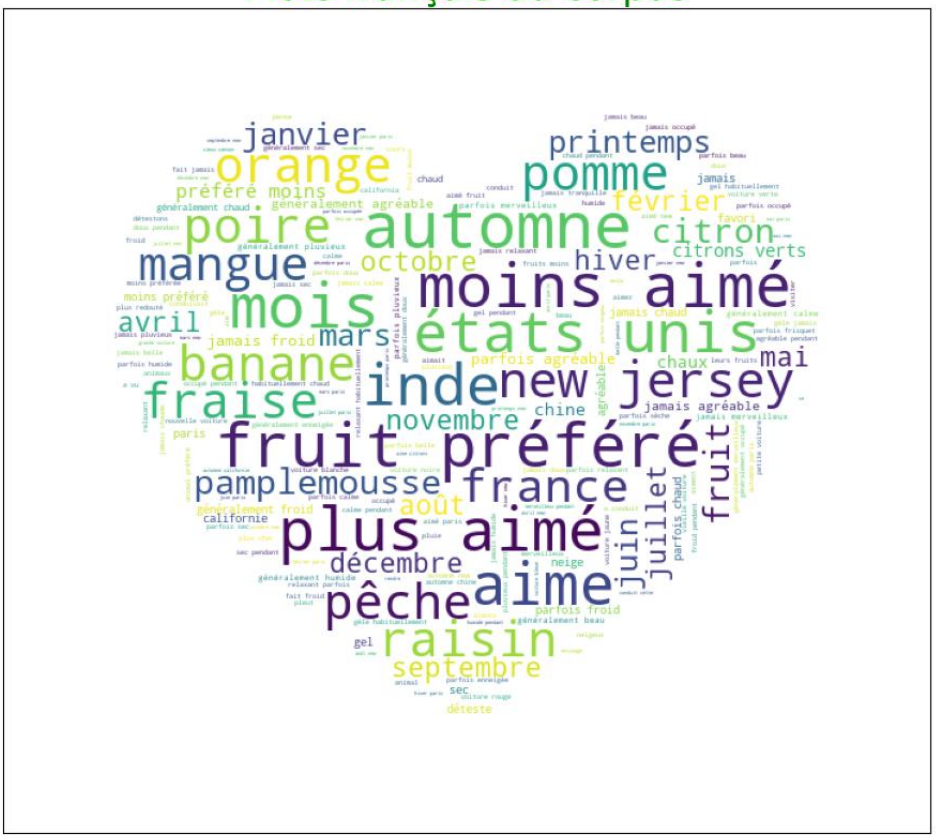
Référence
- *https://ai.facebook.com/blog/wikimatrix/*
- *https://opus.nlpl.eu/WikiMatrix.php*
- *Holger Schwenk, Vishrav Chaudhary, Shuo Sun, Hongyu Gong and Paco Guzman, WikiMatrix: Mining 135M Parallel Sentences in 1620 Language Pairs from Wikipedia, arXiv, July 11 2019.*

Ces données sont en accès libre sur github ou sur le site opus.nlpl.eu

English words corpus



Mots français du corpus



b. Exploration du jeu de données

Notebook : *2, 3 et 5 - Exploration, Preprocessing, Visualisation, et Traduction mot à mot de small_vocab.ipynb*

- **Variables pertinentes**

Les datasets sont composés de phrases et de mots, donc aucune variable ne nous est explicitement fournie. Voici les premières phrases des corpus:

```
EN:  new jersey is sometimes quiet during autumn , and it is snowy in april .
FR:  new jersey est parfois calme pendant l' automne , et il est neigeux en avril .

EN:  the united states is usually chilly during july , and it is usually freezing in november .
FR:  les états-unis est généralement froid en juillet , et il gèle habituellement en novembre .

EN:  california is usually quiet during march , and it is usually hot in june .
FR:  california est généralement calme en mars , et il est généralement chaud en juin .

EN:  the united states is sometimes mild during june , and it is cold in september .
FR:  les états-unis est parfois légère en juin , et il fait froid en septembre .
```

Néanmoins, lors de notre pré-traitement, nous produirons de nombreuses variables intermédiaires nécessaires au processus de traduction :

- **Texte « propre »** : *txt_en, txt_fr.* Texte notamment sans ponctuation.

- **Token** : *txt_split_en, txt_split_fr.*
  Ces variable sont des tableaux numpy à 2 dimensions (dim 1 = phrase, dim 2 = 'mot' dans la phrase) :

[['new', 'jersey', 'is', 'sometimes', 'quiet', 'during', 'autumn', 'and', 'it', 'is', 'snowy', 'in', 'april'],
 ['the', 'united', 'states', 'is', 'usually', 'chilly', 'during', 'july', 'and', 'it', 'is', 'usually', 'freezing', ...], ...]

- **Bag Of Words (BOW)** : *df_count_word_en, df_count_word_fr.*
  Ces dataframes comprennent les phrases en ligne, et les mots en colonnes. Les valeurs sont le nombre d'occurrence du mot dans la phrase (comptage des tokens)

	a	am	and	animal	animals	apple	apples	april	are	aren	...	when	where	white	why	winter	wonderful	would	yellow	you	your
0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
137855	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
137856	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
137857	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
137858	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
137859	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Ainsi, on constate que les mots 'and' et 'april' apparaissent 1 fois dans la 1ere phrase du corpus anglais.

- **Mots uniques (keywords)** : *corpus_en ; corpus_fr* (improprement appelé ainsi) :

```
['new', 'jersey', 'is', 'sometimes', 'quiet', 'during', 'autumn', 'and', 'it',
'snowy', 'in', 'april', 'the', 'united', 'states', 'usually', 'chilly', 'july',
'freezing', 'november', 'california', 'march', 'hot', 'june',…]
```

- **Mots transparents** (stop words)
  Ces mots sont les pluscourants dans la langue et portent moins de signification que les autres. Ces mots sont enlevés des phrases lors du pré-processing
  Exemple anglais : {'a', 'about', 'above', 'after', 'again', 'against',...}

- **Longueur des phrases** (en mots) : *sent_len_en, sent_len_fr* ('sent' signifie « sentence ») :
  sent_len_en = [13, 15, 13, 14, 14, 12, 12,...]

  Ainsi la 1ere phrase anglaise comporte 13 mots, la 2eme 15, etc.

- **Modèle Word Embedding** : *en_model, fr_model*
  Ces modèles nous permettront de convertir les mots en vecteur de 300 dimensions. Les espaces vectoriels dans les lesquels sont « plongés » ces mots, sont alignés. Ainsi des mots de même signification dans 2 langues différentes doivent avoir des vecteurs proches.

- **Variables cible**

Les variables cibles sont des **DataFrames** « **dictionnaire** », comprenant les mots du langage source en nom de colonne, et la traduction de ces mots en 1ere ligne.
Voici la liste des dictionnaires produits avec les différents algorithmes :

- *dict_FR_EN* et *dict_EN_FR* créés avec un BOW et l'algorithme K-Means
- *knn_dict_FR_EN* et *knn_dict_EN_FR* créés avec un BOW et l'algorithme KNN avec k=1
- *rf_dict_FR_EN* et *rf_dict_EN_FR* créés avec un BOW et l'algorithme Random Forest
- *we_dict_FR_EN* et *we_dict_EN_FR* créés avec Vectors-Wiki et la méthode most_similar des modèles de Word Embedding

- **Particularités**

Les corpus de phrases multilingues sont faciles d'accès et nombreux. **Small_vocab** constitue un jeu de données d'une simplicité extrême. Néanmoins ils nécessitent un pré-processing important afin de pouvoir être compris par les algorithmes de traitement.

c. Pre-processing et feature engineering

Notebook :
2, 3 et 5 - Exploration, Preprocessing, Visualisation, et Traduction mot à mot de small_vocab.ipynb

Nettoyage
La première étape est de nettoyer le texte afin de travailler un texte «propre».

- Mettre le texte en minuscule
- Enlever les urls, la ponctuation ("." , "," , ";" , ":" , "?" , "-"), les chiffres et les espaces superflus.
- Corriger les fautes d'orthographe (étape non exécutée car longue et inutile sur ce corpus)
- Lemmatiser, c'est-à-dire remplacer les mots par leur forme neutre canonique que l'on trouve dans un dictionnaire. Ainsi en anglais, le lemme du mot 'is' est 'be', et celui du mot 'bananas' est 'banana'.
- Enlever les stop words (Cf. ci-dessus)
- Il aurait fallu aussi identifier les noms propres (ce que nous n'avons pas fait)
  Note : Dans la suite du projet, nous n'utiliserons pas le texte lemmatisé, ni le texte sans stop words, car nous allons effectuer une traduction mot à mot, et nous avons besoin de tous les mots sous leur forme originelle.

Tokenisation
Le texte est segmenté en mots que nous appellerons "tokens". Nous n'avons pas eu besoin de tokeniser en N-Gram, c'est-à-dire une combinaison de N mots. Nous verrons dans la partie 3, que nous pouvons utiliser d'autre type de token (**Tiktoken** d'OpenAI et **BERT** de Google).

Parts of Speech (POS tagging)
Ce processus traite une séquence de mots et attache un marqueur de parties du discours à chaque mot. La bibliothèque 'nltks poc_tag' est utilisée pour l'étiquetage POS. Voici quelques exemples de balises POS : VBZ -> Verbe, NN-> Nom, PRP -> Préposition, IN -> Interjection.

Bien que nous ayons réalisé ce processus pour l'anglais, il ne nous a pas été utile à ce stade.

Text To Numeric (Term To Digit)
À ce stade du projet et avec les algorithmes que nous avons utilisés, nous n'avons pas eu besoin de convertir les mots en nombre.

Création d'un Bag Of Words (BOW) ou Vectorization TF-IDF
Une fois le texte nettoyé et tokenisé, nous pouvons compter les mots dans chaque phrase et ainsi créer les variables BOW (Cf. ci-dessus), grâce à la méthode CountVectorizer de Sklearn. En fait, nous avons utilisé une forme particulière du BOW dans laquelle on enregistre que la présence du mot dans la phrase (mot présent = 1, absent = 0).

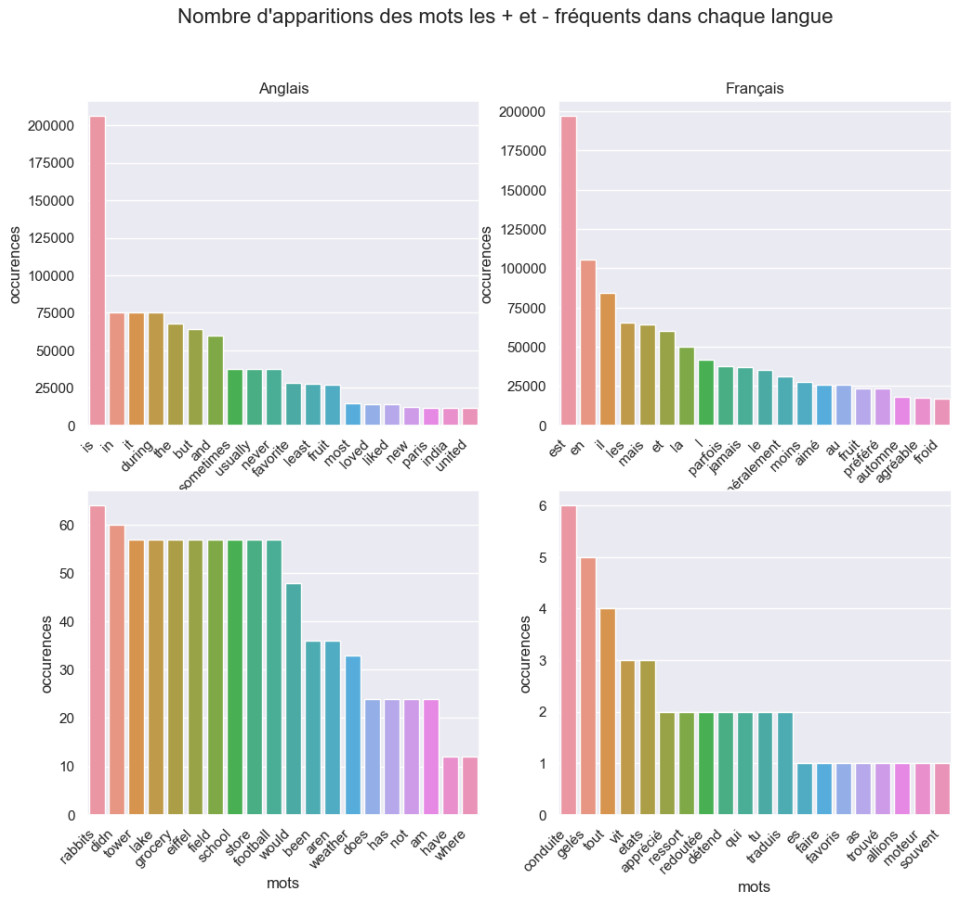
Nous avons aussi réalisé une vectorisation TF-IDF, grâce à la méthode TfidfVectorizer de Sklearn. TF-IDF signifie "Term Frequency-Inverse Document Frequency". Il mesure l'importance d'un terme (mot) par rapport à un document (phrase) dans un corpus. Chaque terme d'un document se voit attribuer un poids après avoir multiplié sa fréquence de terme (tf) et sa fréquence inverse de document (idf). Lors de son utilisation, cette vectorisation n'a pas donné de meilleurs résultats que le BOW. Nous l'avons donc abandonné.

3. Visualisations et Statistiques

Notebook :
2, 3 et 5 - Exploration, Preprocessing, Visualisation, et Traduction mot à mot de small_vocab.ipynb

a. Relation entre les mots

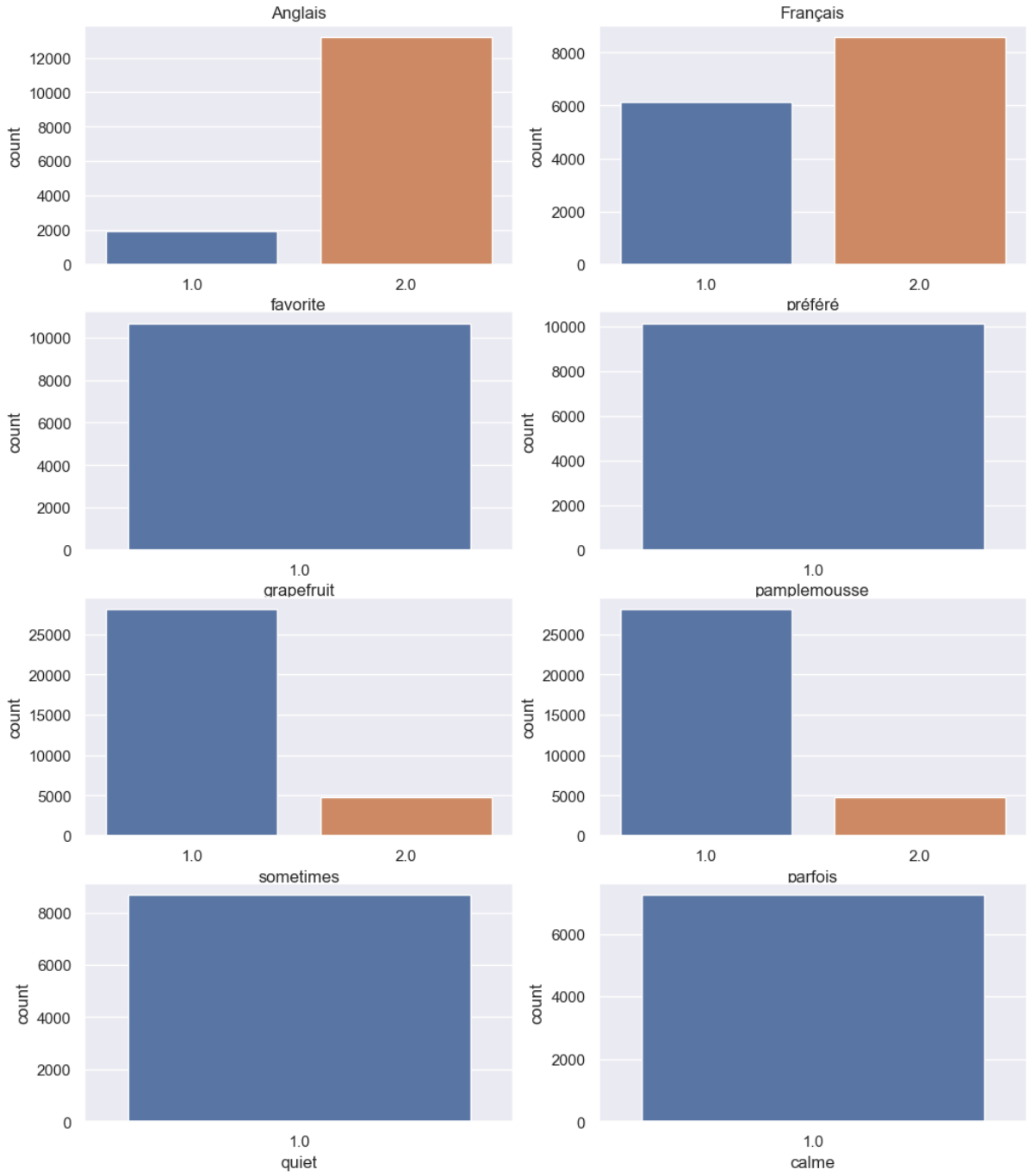
Si nous regardons le nombre d'apparitions des mots dans le corpus, nous voyons une certaine analogie entre les corpus anglais et français, notamment pour les mots les plus fréquents (graphes ci-dessous de la 1er ligne).



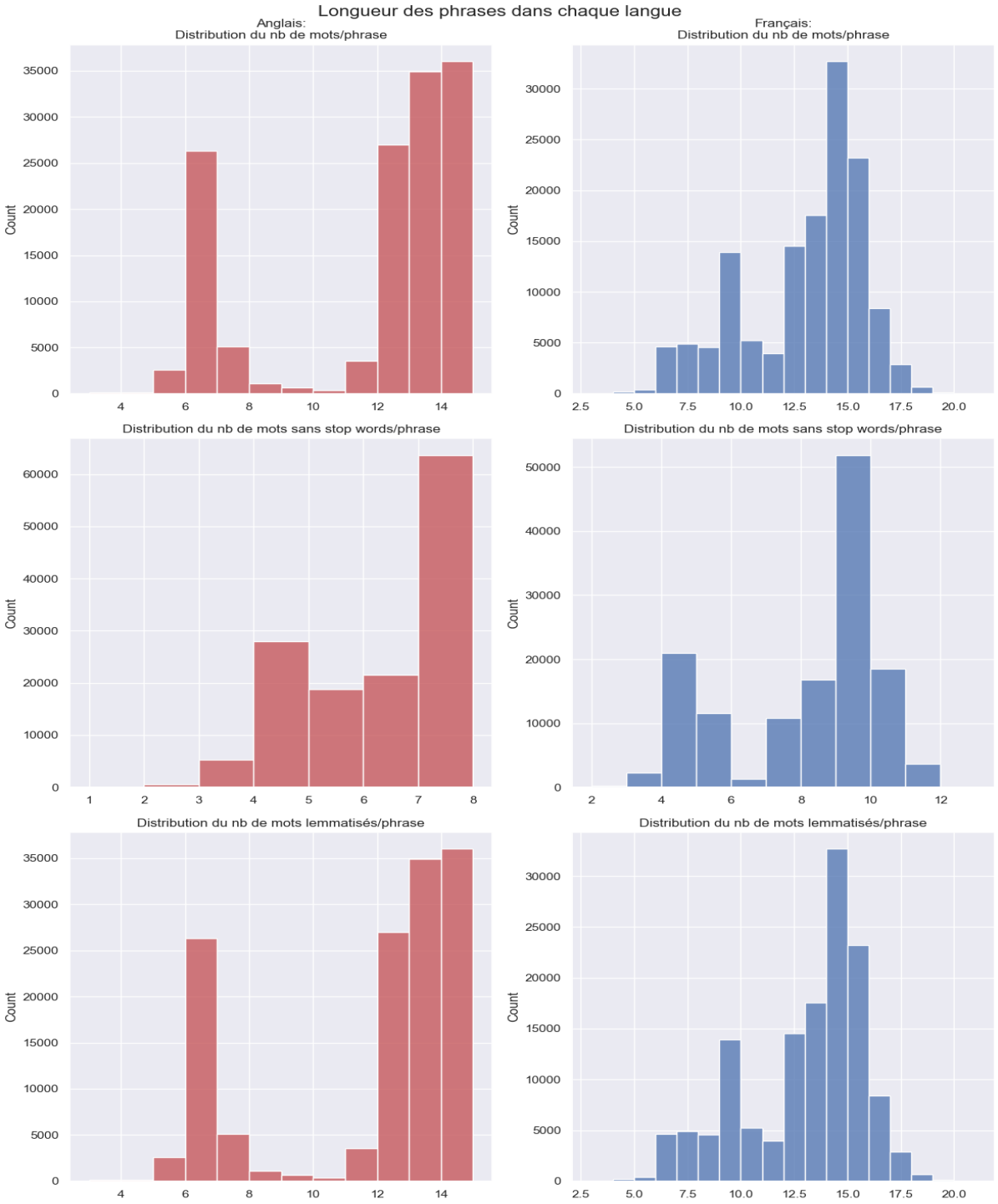
Ainsi, la fréquence d'apparition pourrait permettre d'associer un mot avec sa traduction.

Voici la distribution de fréquence d'apparition dans une phrase de certains mots sélectionnés : 'favorite' et 'préféré' (respectivement en anglais et français), 'grapefruit' et 'pamplemousse', 'sometimes' et 'parfois', 'quiet' et 'calme'. On constate une similitude dans les distributions anglaises et françaises.

Nombre de phrases (y) où l'on trouve certains mots avec une certaine occurence (x)



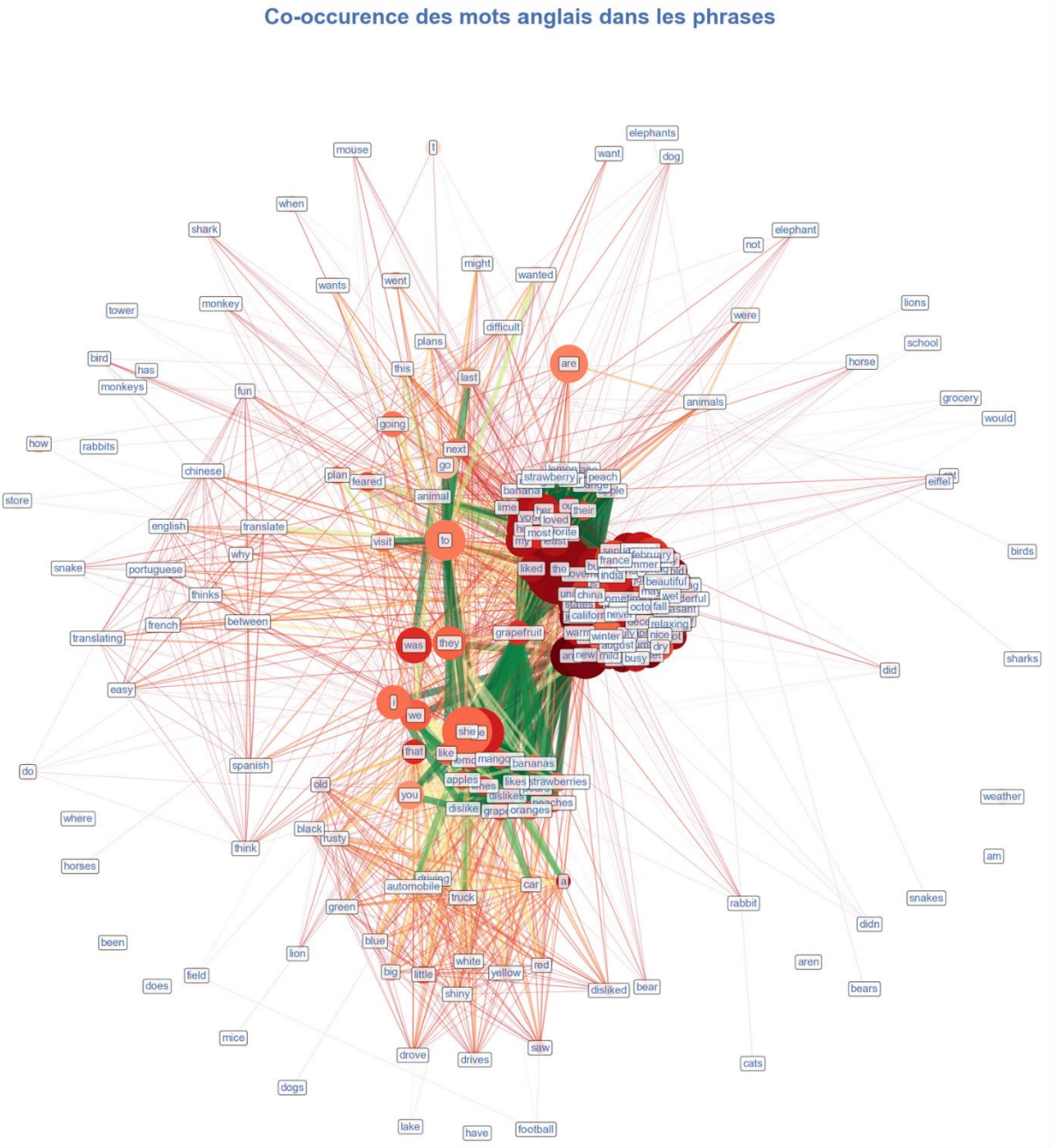
b. Analyse des longueurs de phrase en mot



On constate une certaine similitude entre les 2 langues, même si les phrases françaises ont plus de mots que les phrases anglaises. Cela nous conforte dans l'idée d'utiliser le BOW pour associer les mots des 2 langues

c. Analyse de co-occurrences de mots dans une phrase

Le graphe est difficilement lisible. On constate, bien sûr, que chaque mot apparaît avec beaucoup d'autres mots du corpus.



Exemple sur le mot anglais « fruit » :
Ce mot apparaît dans 23 500 phrases sur les 137 860 du corpus. Voici les premières phrases dans lesquelles ce mot apparaît.

text_en:	your least liked fruit is the grape
Tokens & Tags:	least | like | fruit | grape
text_en:	his favorite fruit is the orange
Tokens & Tags:	favorite | fruit | orange
text_en:	our least liked fruit is the lemon
Tokens & Tags:	least | like | fruit | lemon
text_en:	the lime is her least liked fruit
Tokens & Tags:	lime | least | like | fruit
text_en:	he dislikes grapefruit
Tokens & Tags:	dislike | grapefruit
text_en:	her least liked fruit is the lemon
Tokens & Tags:	least | like | fruit | lemon
text_en:	their favorite fruit is the mango
Tokens & Tags:	favorite | fruit | mango
text_en:	the grapefruit is my most loved fruit
Tokens & Tags:	grapefruit | love | fruit
text_en:	the orange is her least liked fruit
Tokens & Tags:	orange | least | like | fruit
text_en:	the lemon is my most loved fruit
Tokens & Tags:	lemon | love | fruit
text_en:	the apple is our least favorite fruit
Tokens & Tags:	apple | least | favorite | fruit

d. Analyse du Word Embedding des mots des 2 corpus

Nous avons récupéré les vecteurs de chaque mot des Corpus dans « Vectors-Wiki ». Si l'on affiche une « Analyse en Composante Principale » des mots les plus fréquents des 2 corpus, on constate un certaine « proximité » entre les mots anglais et leur traduction.

Modalités d'évaluation :

- Nous avons vu que l'utilisation d'un BOW pour la traduction mot à mot était prometteuse.
- Cependant, il sera intéressant d'étudier l'utilisation du Word Embedding, qui présente l'avantage d'être indépendant des corpus de mots.

4. Identification de langue

a. Jeux de données utilisés

Notebook : *4-1 - Detection de langue sur small_vocab avec BERT rev2.ipynb*

Nous avons commencé par utiliser **small_vocab** avant d'éprouver notre approche sur un jeu plus complet. Ce challenge finalement assez simple nous a permis d'obtenir une accuracy = 100% sur notre test set.

Classe prédite	en	fr
Classe réelle		
en	41358	0
fr	0	41358

Il était donc nécessaire de mettre la barre un peu plus haut. Nous avons par conséquent utilisé un jeu de donnée se trouvant sur **Kaggle**, contenant plus de 10 millions de phrases, dans 404 langues.

Nous avons rebaptisé ce jeu, **sentences-big.csv**, dont nous avons extrait un sous-ensemble de 1 750 000 phrases dans les 5 langues d'Europe de l'Ouest (deu, eng, fra, ita, spa): **sentences.csv** afin de respecter la contrainte de taille de stockage, avec un maximum de 100 Mo sur Github.

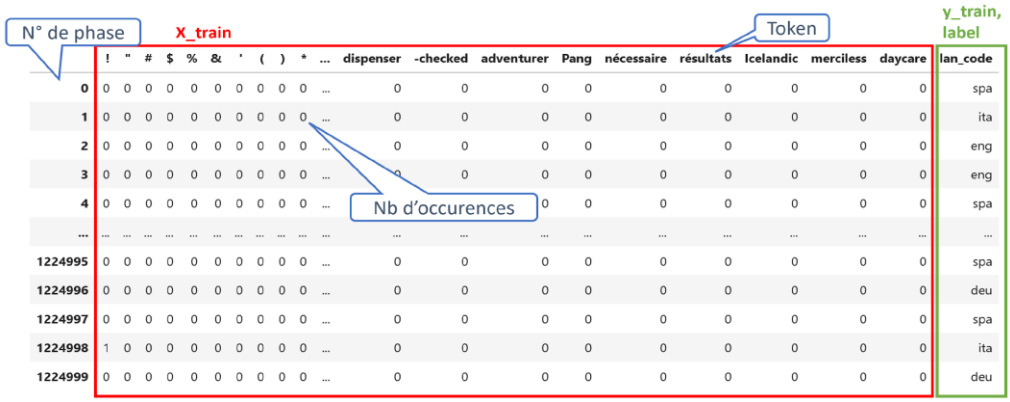
b. Choix du modèle

Nous avons réalisé un détecteur de Langue pour les 5 langues d'Europe de l'Ouest, de 2 manières : un BOW et un modèle de Deep Learning

I. Identification avec la méthode BOW

Notebook : *4-2 - Detection de langue avec BOW, Interpretabilité rev3.ipynb*

L'idée est de définir un BOW avec les tokens comme caractéristiques (features), et le code langue comme label.



	!	"	#	$	%	&	'	(	)	*	...	dispenser	-checked	adventurer	Pang	nécessaire	résultats	Icelandic	merciless	daycare	lan_code
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	spa
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	ita
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	eng
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	eng
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	spa
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1224995	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	spa
1224996	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	deu
1224997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	spa
1224998	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	ita
1224999	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	deu

Nous avons rencontré de nombreuses difficultés dans cette tâche. En effet, la tokenisation “par mot” génère un nombre considérable de tokens. Or, la taille du BOW est proportionnelle au (Nombre de phrases x Nombre de tokens). Ce qui en conclusion, nous a amené à atteindre rapidement le maximum de mémoire de nos ordinateurs.

Nous avons donc opté pour d’autres tokenisation : **BERT** de Google et **Tiktoken** d’OpenAI. Ces 2 tokenisations se sont avérées efficaces. Par la suite, nous avons retenu Tiktoken qui nous a semblé un peu plus pratique (pas de token ‘[CLS]', '[SEP]’, par exemple).

Malgré ce choix, et la diminution du nombre de phrases ‘train’ à 140 000, nous avions encore 40 000 tokens, ce qui était encore trop important.

Nous avons alors choisi de ne garder que les 2000 tokens les plus fréquents, et d’ignorer approximativement 33 000 tokens, soit 94% d’entre eux.

Deux algorithmes de classification supervisés ont ensuite été implémentés :
- Naives-Bayes
- Gradiant Boosting

De manière, assez surprenante, nous avons obtenu une **accuracy** respective de **99,2%** et **97,2%** sur l’ensemble test.

Naïve Bayes

Classe prédite / Classe réelle	deu	eng	fra	ita	spa
deu	27944	11	9	20	10
eng	50	27947	21	70	52
fra	14	9	27835	86	127
ita	21	10	53	27486	211
spa	29	31	104	192	27658

```
Accuracy Naïve Bayes = 0.992
```

Gradiant Boosting

Classe prédite / Classe réelle	deu	eng	fra	ita	spa
deu	27341	163	79	149	262
eng	67	27697	13	58	305
fra	31	53	27212	298	477
ita	160	107	177	26662	675
spa	63	95	225	474	27157

```
Accuracy Gradiant Boosting = 0.972
```

1ere amélioration, créer un tokenizer personnalisé

Avec la sélection des tokens les plus fréquents - au-delà du fait d'avoir à calculer leur fréquence pour les sélectionner - nous perdons beaucoup d'informations.

Suite à de nombreux essais, nous avons pu utiliser **CountVectorizer** en injectant BERT ou Tiktoken comme 'custom tokenizer' (avec un 'custom preprocessor' identité), apportant de nombreux bénéfices :

- CountVectorizer génère une matrice creuse (Sparse Matrix), ce qui signifie que l'on peut conserver tous les tokens (43 361)
- On peut traiter désormais tout le corpus de phrases 'train' (1 225 000 versus 140 000 précédemment)
- CountVectorizer est beaucoup plus rapide que la fonction que nous avions développé pour créer le BOW (qui sélectionnait les tokens les plus fréquents)

Résultats : L'accuracy est passé respectivement à **99,8%** et **97,1%** avec les 2 méthodes (pas d'amélioration avec Gradient boosting, mais sur un plus grand nombre de phrases).

Naïve Bayes

Matrice de confusion du classificateur Naïve Bayes

Classe prédite / Classe réelle	deu	eng	fra	ita	spa
deu	105217	6	10	19	11
eng	43	105004	25	47	45
fra	6	5	104340	95	93
ita	5	6	28	104680	218
spa	16	3	61	252	104765

Accuracy Naïve Bayes = 0.998

Gradiant Boosting

Matrice de confusion du classificateur Gradiant Boosting

Classe prédite / Classe réelle	deu	eng	fra	ita	spa
deu	102926	546	317	459	1015
eng	229	103475	55	162	1243
fra	142	192	101329	1075	1801
ita	705	272	744	100333	2883
spa	227	408	936	1555	101971

Accuracy Gradiant Boosting = 0.971

2eme amélioration, utilisation du Naïve Bayes sur un jeu plus complet

Notebook : [4-3-a - Detection de 95 langues avec BOW et NB rev4.ipynb]

Libéré de cette contrainte de taille, nous avons appliqué l'algorithme Naïve Bayes de meilleure accuracy, au corpus **sentences-big** de plus de 10 Millions de phrases.

Pour un travail propre et efficace, nous n'avons conservé que les langues dont nous avions au moins 2 000 phrases, passant ainsi de 404 à 95 langues, et représentant 10.345.978 de phases, soit seulement 12 000 phrases en moins.

Résultat : L'Accuracy = **96% sur les 95 langues**

Matrice de confusion du classificateur Naïve Bayes

Classe prédite / Classe réelle	afr	ara	arq	asm	avk	aze	bel	ben	ber	bre	...	uig	ukr	urd	vie	vol	war	wuu	yid	yue	zsm
afr	149	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
ara	0	1923	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
arq	0	103	0	0	0	0	0	0	2	0	...	1	0	0	0	0	0	0	0	0	0
asm	0	0	0	110	0	0	0	63	0	0	...	0	0	0	0	0	0	0	0	0	0
avk	0	0	0	0	180	0	0	0	3	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
war	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	73	0	0	0	0
wuu	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	75	0	2	0
yid	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	420	0	0
yue	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	208	0
zsm	0	6	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	92

95 rows × 95 columns

Accuracy Naïve Bayes = 0.960

Ce résultat est encore amélioré si nous ne regardons que les 5 langues des 5 pays d'Europe de l'Ouest, nous obtenons un **accuracy = 99,1%**

Classe prédite / Classe réelle	deu	eng	fra	ita	spa	etc
deu	29330	4	1	1	0	39
eng	11	78916	3	7	8	182
fra	8	8	24833	26	13	49
ita	1	2	22	40340	38	173
spa	1	1	4	35	18321	147
etc	85	46	102	332	447	320029

```
Accuracy pour les langues d'Europe de l'Ouest = 0.991
```

Exemples d'identification

Avec des phrases issues du test set ou imaginées par l'équipe, dont une phrase piège contenant plusieurs langues.

```
# Instanciation d'un exemple
exemples = ["Er weiß überhaupt nichts über dieses Buch.",                                                    # Phrase 0
            "france is often snowy during spring , and it is relaxing in january .",                         # Phrase 1
           "elle adore les voitures très luxueuses, et toi ?",                                               # Phrase 2
           "she loves very luxurious cars, don't you?",                                                      # Phrase 3
           "vamos a la playa",                                                                               # Phrase 4
           "Ich heiße Keyne, und das ist wunderbar",                                                         # Phrase 5
           "she loves you much, mais elle te hait aussi and das ist traurig.", # Attention à cette phrase trilingue  # Phrase 6
           "A crane raises heavy construction materials.",                                                   # Phrase 7
           "Vogliamo visitare il Colosseo e nuotare nel Tevere.",                                            # Phrase 8
           "私はそれについて全く知りません"                                                                    # Phrase 9
          ]
Langue réelle               : ['deu', 'eng', 'fra', 'eng', 'spa', 'deu', 'en,fr,de', 'en', 'ita', 'jpn']
Prédictions Naive Bayes     : ['German', 'English', 'French', 'English', 'Spanish', 'German', 'Galician', 'English', 'Italian', 'Japanese']
```

II. Identification avec la méthode Deep Learning

1ere tentative

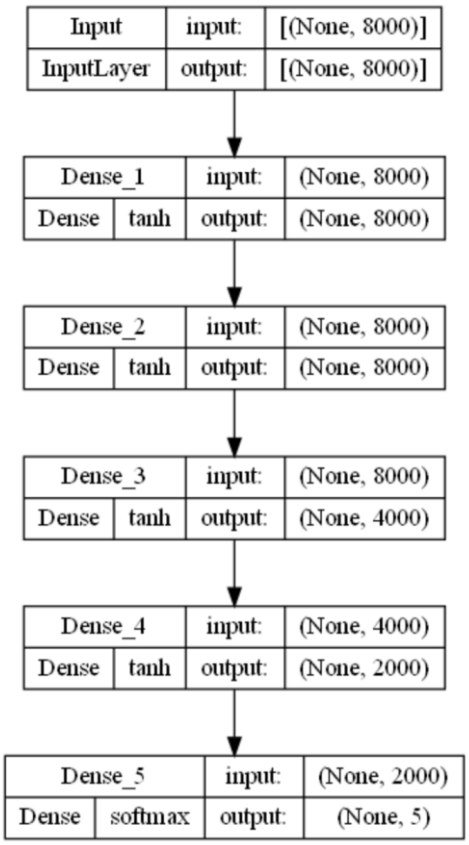
Notebook : *4-4 - Detection de langue avec le Deep Learning et BOW rev1.ipynb*

Nous avons réalisé notre première tentative de détection de langue d'Europe de l'Ouest, en injectant un BOW de 8000 tokens et 28 000 lignes en input dans notre modèle.

Nous ne pouvions pas augmenter les dimensions de notre BOW, en raison de sa taille. Or, il semble beaucoup plus difficile d'utiliser un modèle de Deep Learning sur une matrice creuse (ou un dataframe creux).

Modèle

:



Résultats

Les résultats sur le 'test set' sont bons mais limités :

	precision	recall	f1-score	support
0	0.99	1.00	0.99	2452
1	0.99	1.00	0.99	2399
2	0.99	0.98	0.98	2395
3	0.99	0.97	0.98	2381
4	0.96	0.99	0.97	2373
accuracy			0.98	12000
macro avg	0.98	0.98	0.98	12000
weighted avg	0.98	0.98	0.98	12000

Classe prédite	**0**	**1**	**2**	**3**	**4**
Classe réelle					
0	2441	4	2	3	2
1	3	2389	0	2	5
2	3	20	2336	7	29
3	8	3	10	2309	51
4	9	8	5	11	2340

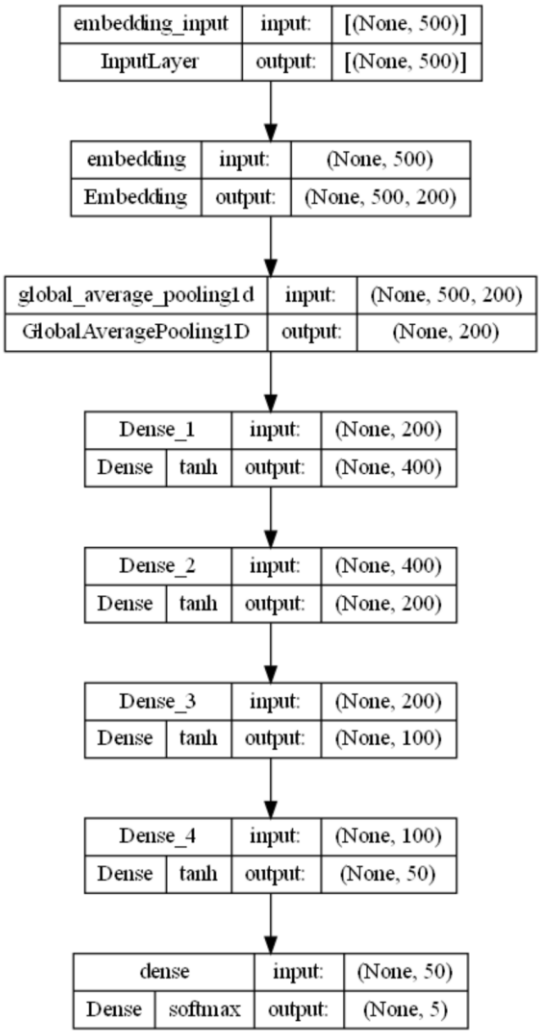
Accuracy Classifier = 0.985

2eme tentative

Afin de supprimer les limites rencontrées, nous avons transformé les entrées en vecteur d'embedding (dimension des vecteurs =200) pour des phrases de 500 tokens max. Les dimensions (500 x 200 floats) sont tout à fait acceptables par nos ordinateurs.

À la suite, nous avons ajouté une couche de « GlobalAveragePooling » pour réduire la dimension de la couche d'embedding et pouvoir y connecter les couches denses suivantes.

Modèle



Résultats

(sur les 5 langues d'Europe de l'Ouest)

	precision	recall	f1-score	support
deu	1.00	1.00	1.00	105263
eng	1.00	1.00	1.00	105164
fra	1.00	1.00	1.00	104539
ita	1.00	1.00	1.00	104937
spa	1.00	1.00	1.00	105097
accuracy			1.00	525000
macro avg	1.00	1.00	1.00	525000
weighted avg	1.00	1.00	1.00	525000

======

Classe prédite / Classe réelle	deu	eng	fra	ita	spa
deu	105158	69	10	17	9
eng	6	105129	7	12	10
fra	4	102	104356	26	51
ita	5	67	23	104720	122
spa	6	83	21	177	104810

Accuracy Classifier = 0.998

3eme tentative

Notebook : [4-5 - Detection de langue avec le Deep Learning et Embedding rev3.ipynb]

Nous pouvons désormais aborder le dataset plus complet **sentences-big** et ses **95 langues**. Nous pourrons ainsi comparer les résultats d'un BOW et un classificateur Naïve Bayes, avec un modèle de Deep Learning contenant 4 couches denses.

Classe prédite / Classe réelle	afr	ara	arq	asm	avk	aze	bel	ben	ber	bre	...	uig	ukr	urd	vie	vol	war	wuu	yid	yue	zsm
afr	182	0	0	0	0	0	0	0	1	0	...	0	0	0	0	1	0	0	0	0	0
ara	0	1912	9	0	0	0	0	0	1	0	...	1	0	0	0	0	0	0	0	0	0
arq	0	34	73	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
asm	0	0	0	146	0	0	0	27	0	0	...	0	0	0	0	0	0	0	0	0	0
avk	0	0	0	0	158	0	0	0	3	3	...	0	0	0	0	0	1	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
war	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	92	0	0	0	0
wuu	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	200	0	5	0
yid	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	440	0	0
yue	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	2	0	288	0
zsm	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	199

95 rows × 95 columns

```
Accuracy Classifier = 0.975
```

Classe prédite / Classe réelle	deu	eng	fra	ita	spa	etc
deu	29225	10	5	3	1	131
eng	2	79035	8	6	6	70
fra	1	4	24864	17	6	45
ita	0	1	10	40442	20	103
spa	0	0	4	44	18322	139
etc	39	84	104	302	358	320154

```
Accuracy pour les langues d'Europe de l'Ouest = 0.992
```

Exemples d'identification :

À la suite de ces résultats, nous avons implémenté des exemples de prédictions

```
Exemples de prédiction de langue:
Réelle  - Prédite - Texte
 ita    - ita     -'Sto leggendo il libro 'La lingua pericolosa' di Ulrich Lins.....'
 lit    - lit     -'Taip, tai yra labai gražu. Kiek tai kainuoja?.....'
 eng    - eng     -'The man gave a big cry.....'
 tur    - tur     -'Burada bekle. Ben kısa zamanda dönerim.....'
 ita    - ita     -'Perché mi sospetti?.....'
 por    - por     -'Não sei a quem o darei.....'
 rus    - rus     -'Отведите их туда.....'
 tur    - tur     -'Beyazdı.....'
 kab    - kab     -'Bubbent ar At Wuccen.....'
 spa    - spa     -'Los atracadores acribillaron a tiros a los agentes que se aproximaron al edificio.....'
```

Exemple de mauvaises prédictions :

```
Exemples de mauvaises prédictions de langue:
Réelle  - Prédite - Texte
 deu    - dan     -'Hilft Tom Mary?.....
 ber    - kab     -'Iruka-a n yemma.....
 kab    - ber     -'Lliɣ ttnaɣeɣ d yiḍ, tawla, iḍes asemmiḍ.....
 por    - glg     -'É esta a chave que estás buscando?.....
 slk    - ces     -'Je odo mňa o hodne vyššia.....
 ota    - uig     -'اونڭ بابه سى وار.....
 kab    - ber     -'Σerḍeɣ-t-id maca ɣas akken, ur d-yusa ara...
 ukr    - rus     -'Том одягнув шкарпетки.....
 bul    - mkd     -'На кого е това палто?.....
 rus    - ukr     -'Я принимаю наркотики.....
```

Conclusion sur la performance de l'identification de langue

Plus simple à implémenter, le modèle de Deep Learning donne légèrement de meilleurs résultats, mais a besoin, néanmoins, d'une longue durée d'entraînement. Le temps de prédiction est aussi plus long.

	BOW	DL
95 langues	96,00%	97,50%
5 langues d'Europe de l'Ouest	99,10%	99,20%

c. Interprétabilité de l'algorithme Naïve Bayes avec BOW

Notebook : *4-2 - Detection de langue avec BOW, Interpretabilité rev3.ipynb*

Il nous a semblé intéressant de réaliser un travail d'interprétabilité afin de comprendre plus précisément ce qui permettait à l'algorithme de décider l'attribution d'une langue plutôt qu'une autre.

Cette interprétabilité a été réalisée sur le jeu de données **sentences.csv** avec les 5 langues d'Europe de l'Ouest et 1 750 000 phrases.

Les variables explicatives sont les 43361 tokens issus du tokenizer Tiktoken, et la variable expliquée est la langue prédite (lan_code).

Nous avons utilisé différentes méthodes d'interprétabilités aux résultats plus ou moins exploitables :

- Analyse en Composante Principale
- LIME
- Simulation de l'algorithme Naïve Bayes
- SKATER
- Shapley

Les méthodes Lime et Skater ont donné peu d'explications, peut-être par manque d'expérience.

1. Analyse en Composante Principale

L'ACP permet d'afficher en 2 dimensions, les groupes de phrase par langues, ainsi que l'importance de certains tokens dans ces langues.

Nous n'avons utilisé ici que 400 phrases et affiché les 50 tokens les plus importants. À noter que cette représentation fait perdre beaucoup d'informations, puisque nous projetons 43 000 dimensions sur seulement 2 axes.

La couleur des points affiche le regroupement des phrases par langue (**deu**, **fra**, **eng**, **ita**, **spa**). Nous constatons aussi que certains tokens sont « spécialisés » dans certaines langues, avec par exemple :

- 'Ich', 'nicht', 'die' pour **deu**
- 'the', 'of', 'you' pour **eng**
- 'le', 'de', 'pas', 'que', 'la' pour **fra**

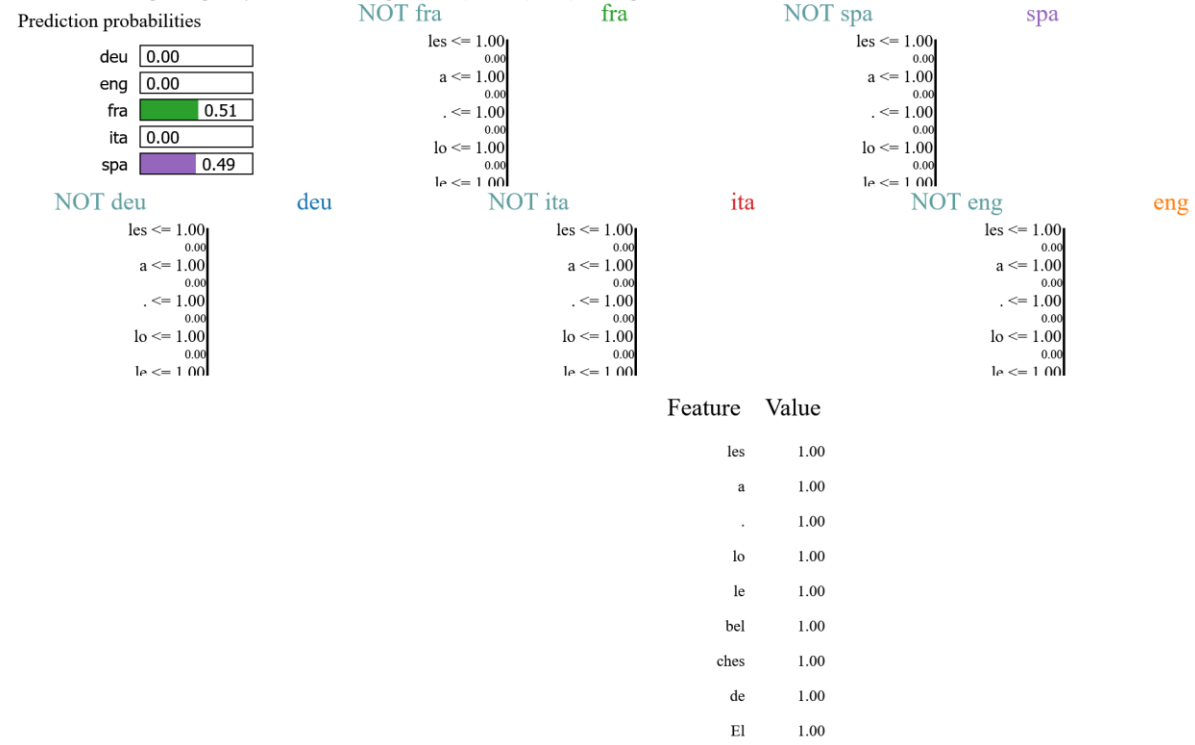
2. LIME

Nous avons utilisé LIME (Local Interpretable Model-agnostic Explanations) sur quelques phrases d'exemple (prédiction locale). Nous avons réduit la taille du BOW afin que les générations de proxies ne soient pas trop importantes et l'algorithme trop lent.

LimeTabularExplainer fut un succès modéré. La probabilité calculée par le modèle semble cohérente, et nous permet de voir qu'elle langue est plus probable qu'une autre. Cependant, cette approche est dans l'incapacité de nous indiquer les tokens qui permettent de calculer cette probabilité.

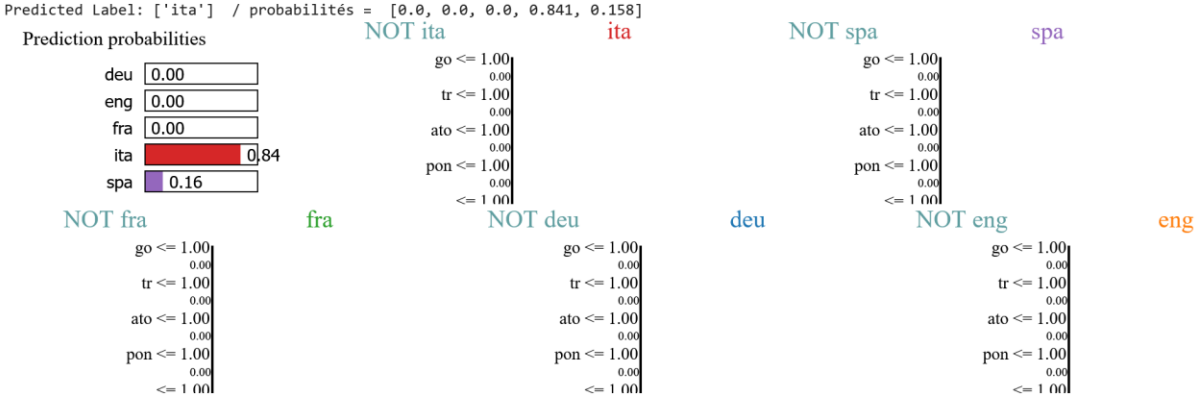
Exemples :

```
Phrase : Elle a de belles loches.
Predicted Label: ['fra']  / probabilités =  [0.0, 0.0, 0.509, 0.0, 0.491]
```



Feature	Value
les	1.00
a	1.00
.	1.00
lo	1.00
le	1.00
bel	1.00
ches	1.00
de	1.00
El	1.00

```
Phrase : Te propongo un trato.
Predicted Label: ['ita']  / probabilités =  [0.0, 0.0, 0.0, 0.841, 0.158]
```



Feature	Value
go	1.00
tr	1.00
ato	1.00
pon	1.00
.	1.00
un	1.00
pro	1.00
Te	1.00

3. Simulation de l'algorithme Naïve Bayes

Quoi de plus efficace que de retourner à la source de l'algorithme pour comprendre son fonctionnement !

Les classificateurs naïfs de Bayes sont une famille de "classificateurs probabilistes" linéaires basés sur l'application du théorème de Bayes avec de fortes hypothèses d'indépendance (naïve) entre les caractéristiques (tokens du BOW).

En résumé, la distribution de probabilité d'une langue pour une phrase qui contient les tokens **F_i**, avec $i \in \{1..n\}$, se calcule ainsi :

$$p(C|F_1,\ldots,F_n) = \frac{1}{Z}p(C)\prod_{i=1}^{n}p(F_i|C)$$

où **C** est la classe Langue (lan_code), **F_i** est la caractéristique **i** du BOW, et **Z** est l' « évidence » servant à régulariser la probabilité .

Étudions la phrase suivante pour illustrer cette formule :
she loves you much, mais elle te hait aussi and das ist traurig.

Cette phrase « piège », habilement conçu par nos soins, est un mélange de trois langues : **eng, fra, deu**

Quelle langue l'algorithme va-t-il alors prédire ?

Étape 1

Décomposons la phrase en 18 tokens, qui influenceront l'algorithme.

```
Nombre de tokens retenus dans le BOW: 18
Tokens retenus (se trouvant dans le modèle) :she loves you much , mais elle te h ait aussi and das ist tr aur ig .
```

Étape 2

Calculons le nombre d'apparitions des tokens dans chaque langue, sans oublier d'appliquer un **lissage de Laplace** (Laplace smoothing) pour les nombres d'apparition = 0. En effet, si un token n'apparait pas dans le BOW, il générera un p(Fi/C) =0, ce qui entraine une probabilité = 0 d'identifier cette langue, puisque l'on multiplie les probabilités. L'astuce consiste à ajouter 1 à toute la matrice (ou tout du moins aux cellules = 0)

Nombre d'apparitions des tokens retenus, dans chaque langue

	1-she	2- loves	3- you	4- much	5-,	6- mais	7- elle	8- te	9- h	10-ait	11- aussi	12- and	13- das	14- ist	15- tr	16-aur	17-ig	18-.
eng	3	273	31145	2601	26724	1	1	47	186	12	1	13217	1	1	62	32	49	216706
fra	1	1	3	1	32086	3620	2237	5087	1202	9846	2013	15	2	1	970	892	722	202787
deu	3	1	2	1	78698	1	1	464	4877	17	1	501	18754	39403	2132	294	9278	202873
spa	1	1	3	863	27970	1	4	6368	3388	10	1	365	97	1	1738	24	1319	207887
ita	1	1	2	1	17099	7	1	2340	16	18	1	5997	1	205	1137	66	1421	189104

Étape 3

Calculons, pour chaque token, la probabilité conditionnelle p(Fi/C) sachant la langue (C) :

Ainsi pour le token 'much' :
p('much'/'eng') = 2601 / 244 847 soit **0,010623**
avec 244 847 le nombre de phrases anglaises dans le BOW

Probabilité d'apparition tokens retenus, dans chaque langue

	1-she	2- loves	3- you	4- much	5-,	6- mais	7- elle	8- te	9- h	10-ait	11- aussi	12- and	13- das	14- ist	15- tr	16-aur	17-ig	18-.	Proba
eng	0.000012	0.001115	0.127208	0.010623	0.109151	0.000004	0.000004	0.000192	0.000760	0.000049	0.000004	0.053983	0.000004	0.000004	0.000253	0.000131	0.000200	0.885107	0.000
fra	0.000004	0.000004	0.000012	0.000004	0.130717	0.014748	0.009113	0.020724	0.004897	0.040112	0.008201	0.000061	0.000008	0.000004	0.003952	0.003634	0.002941	0.826148	0.977
deu	0.000012	0.000004	0.000008	0.000004	0.321562	0.000004	0.000004	0.001896	0.019928	0.000069	0.000004	0.002047	0.076629	0.161001	0.008711	0.001201	0.037910	0.828943	0.023
spa	0.000004	0.000004	0.000012	0.003524	0.114208	0.000004	0.000016	0.026002	0.013834	0.000041	0.000004	0.001490	0.000396	0.000004	0.007097	0.000098	0.005386	0.848854	0.000
ita	0.000004	0.000004	0.000008	0.000004	0.069774	0.000029	0.000004	0.009549	0.000065	0.000073	0.000004	0.024471	0.000004	0.000837	0.004640	0.000269	0.005799	0.771655	0.000
	deu	eng	eng	eng	deu	fra	fra	spa	deu	fra	fra	eng	deu	deu	deu	fra	deu	eng	fra

Étape 4

Multiplions ensuite les probabilités conditionnelles de chaque ligne, multiplié par la probabilité de la langue, p(C).

Par exemple pour l'anglais :
p('eng') = (Nb phrases anglaises dans le BOW/ Nb total de phrases) = 244 844 / 1 225 000 = 19,98 %.

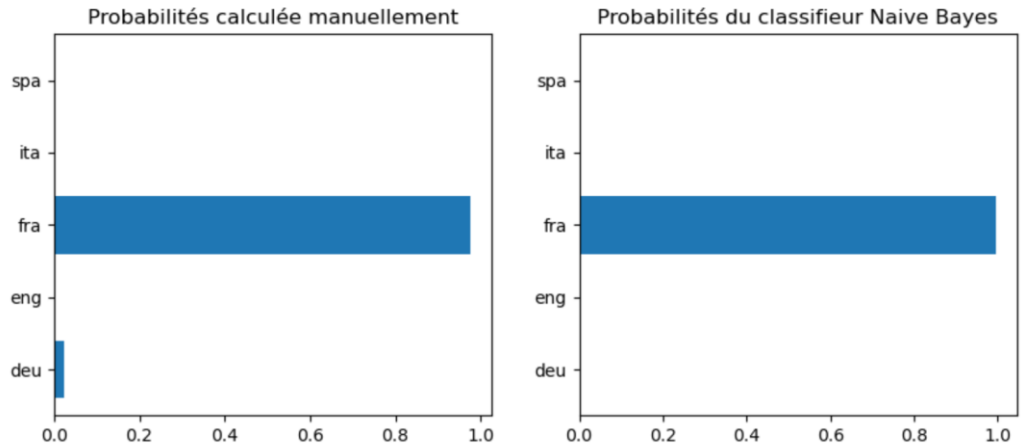
Le résultat final apparait ci-dessus dans la colonne **Proba**.

Sous la matrice ci-dessus, vous pouvez voir aussi pour chaque token, la classe dont la probabilité est la plus forte. Observez, par exemple, qu'il est plus fréquent de trouver 'much' dans la langue anglaise (**eng**) que dans les 4 autres langues...

Conclusion

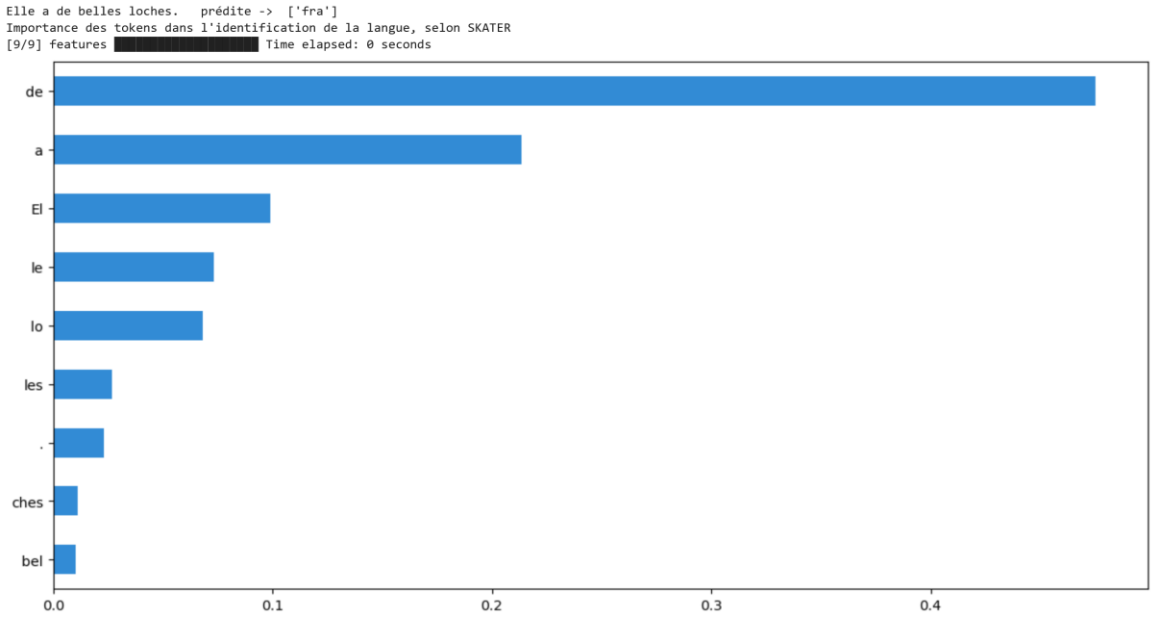
Dans cet exemple, nous constatons un résultat très similaire à celui calculé par l'algorithme Naïve Bayes python. 9 phrases d'exemples sont disponibles dans le notebook..

```
Langue réelle de la phrase                          :  en,fr,de
Langue dont la probabilité est la plus forte        :  fra  (proba=0.98)
Langue dont la probabilité est prédite par Naiva Bayes :  fra  (proba=1.00)
```



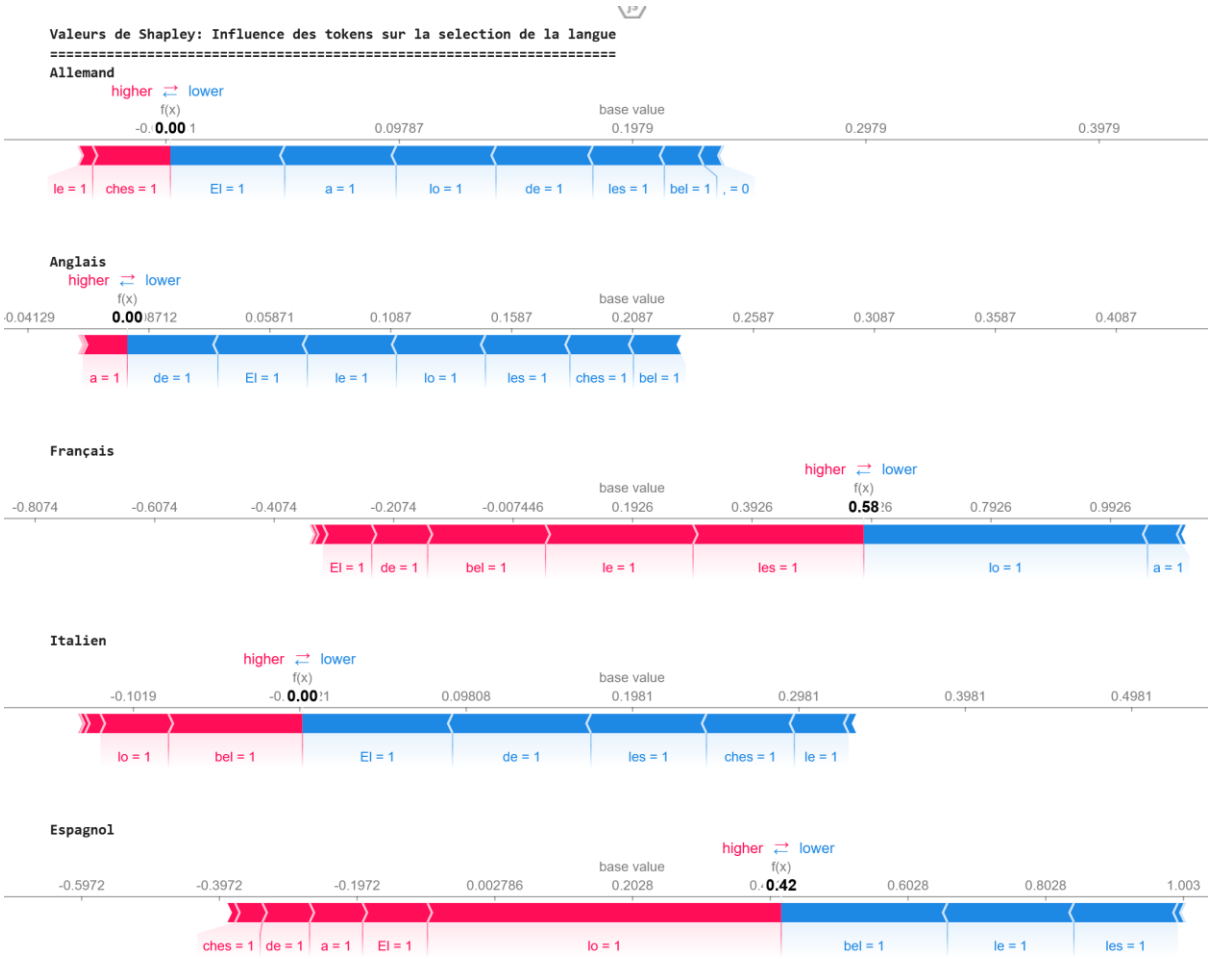
4. SKATER

SKATER nous a permis d'afficher les « feature importances ».
Ainsi, la phrase " **elle a de belles loches** " nous donne le résultat suivant :



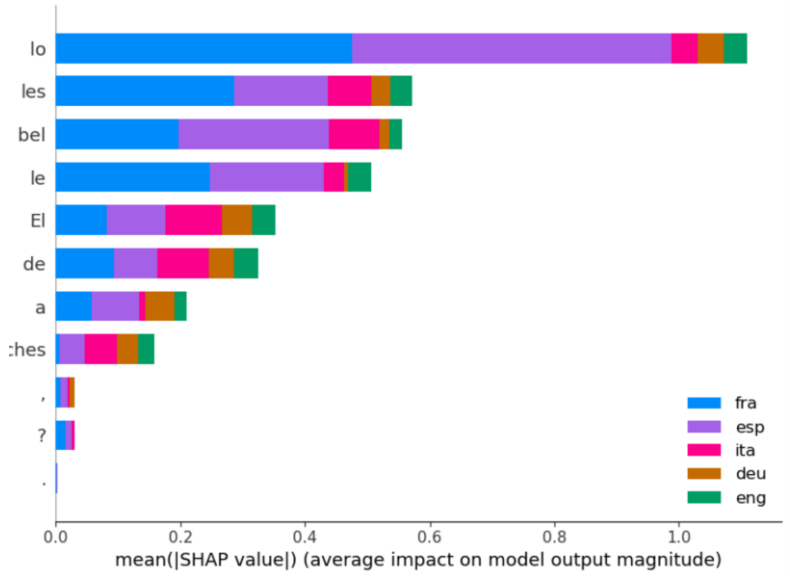
5. Shapley

Avec les valeurs de Shapley, on peut aussi voir les tokens qui ont influencé la classification de l'algorithme.
Exemple : **Elle a de belles loches.**



On constate que cette phrase aurait presque pu être espagnole, chose que nous avions vu avec LIME.

Récapitulatif, avec les valeurs de Shapley, de l'influence des tokens sur la sélection de la langue



5. Traduction mot à mot

Notebook :
2, 3 et 5 - Exploration, Preprocessing, Visualisation, et Traduction mot à mot de small_vocab.ipynb

a. Méthodologie

Dans un premier temps, nous avons abordé la problématique avec une approche naïve de traduction mot à mot. Naturellement, cette approche atteint rapidement des limites. Elle ne prend pas compte d'un certain nombre d'éléments qui font une traduction de qualité, comme la considération des mots environnants ou l'ordre des mots.

Pour cela nous avons créé une Injection entre le corpus anglais de **small_vocab** et le corpus français, et une surjection entre le corpus français et l'anglais.

Rappel : le corpus anglais contient 199 mots et 330 mots en français

L'objectif est d'associer automatiquement un mot de chaque langue. En terme simple, cela signifie réaliser 2 dictionnaires (FR->EN, EN->FR), en utilisant les textes et leur traduction pour la phase d'apprentissage.

Lors d'une première phase, nous allons associer chaque mot d'une langue avec un mot de l'autre langue (catégorie). Cela peut être réalisé par des **algorithmes de classification** (supervisés) **ou de clustering** (non supervisés).

Dans une deuxième phase, nous allons « plonger » nos mots dans une **vectorisation « FasText »**, afin de trouver la traduction d'un mot par similitude de vecteur.

b. Choix de l'algorithme de classification

Pour mesurer la qualité de la traduction des algorithmes de clustering et classification, nous avons constitué un **dictionnaire de référence** « idéal » à la main, dont nous connaissons la meilleure traduction de chaque mot dans le corpus de l'autre langue (dict_FR_EN_ref, dict_EN_FR_ref). Cela permet de définir une fonction « **précision »** de la traduction (% de mots correctement traduits par rapport au dictionnaire de référence).

La labellisation de chaque mot par sa traduction peut être réalisée par des algorithmes de classification (supervisés) ou de clustering (non supervisés).

Clustering non supervisée avec K-Means

Chaque mot représente ici un cluster, labellisé de sa traduction. Les caractéristiques (« features ») sont les vecteurs du BOW transposé, c'est-à-dire des vecteurs ayant une dimension = 137 860, contenant des 0 ou des 1 en fonction de la présence du mot dans une phrase. Ce vecteur constitue une empreinte unique du mot.

Il n'est pas nécessaire d'entraîner longtemps le modèle puisque le centroïde du cluster est le vecteur du mot et que l'on ne souhaite pas modifier le centroïde (1 itération suffit).

Exemple : Ensemble d'entraînement pour un dictionnaire FR ->EN

label	0	1	2	3	4	5	6	7	8	9	...	137850	137851	137852	137853	137854	137855	137856	137857	137858	137859
	X_train																			N° de phrase	
a	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
am	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
and	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0		0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0
animal	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
animals	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
wonderful	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
would	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
yellow	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
you	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
your	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Présence dans ces phrases

Considérant les mots de l'autre langue, avec ses vecteurs BOW comme étant l'ensemble de test, il suffit de prédire le label des mots de la 2eme langue dans la première langue.

Voici le résultat obtenu :

```
Dictionnaire Anglais -> Français:
169  mots corrects /  199
Précision du dictionnaire = 84.92%
```

Anglais	a	am	and	animal	animals	apple	apples	april	are	aren	...	when	where	white	why	winter	wonderful	would	yellow	you	your
Francais	une	vais	et	animal	animaux	pomme	pommes	avril	sont	allions	...	quand	où	blanche	pourquoi	hiver	merveilleux	voudrait	jaune	vous	votre

```
Dictionnaire Français -> Anglais:
152  mots corrects /  330
Précision du dictionnaire = 46.06%
```

Francais	a	agréable	aimait	aime	aiment	aimeraient	aimez	aimons	aimé	aimée	...	à	école	éléphant	éléphants	épicerie	étaient	était	états	été	êtes
Anglais	drove	pleasant	disliked	likes	they	have	you	have	loved	have	...	fall	school	elephant	elephants	grocery	were	was	states	summer	have

Classification supervisée avec K-NN et Random Forest

Là aussi, nous créons une classe 'mot', pour prédire ensuite le label d'un mot de l'autre langue. Dans le cas de l'algorithme du K-NN, on utilise k=1, puisque l'on veut associer un mot cible à un mot source. L'avantage de K-NN sur le K-Means est que l'on peut utiliser une métrique différente de la métrique euclidienne imposée par le K-Means. Dans notre projet, nous avons utilisé la métrique 'minkowski' pour le dictionnaire EN ->FR et la métrique ' cosine' pour le dictionnaire FR->EN (métriques qui donnent la meilleure précision).

Voici les résultats, assez encourageant pour une traduction mot à mot.

K-NN

```
Dictionnaire Anglais -> Français:
171  mots corrects /  199
Précision du dictionnaire = 85.93%
```

Anglais	a	am	and	animal	animals	apple	apples	april	are	aren	...	when	where	white	why	winter	wonderful	would	yellow	you	your
Francais	une	vais	et	animal	animaux	pomme	pommes	avril	sont	allions	...	quand	où	blanche	pourquoi	hiver	merveilleux	voudrait	jaune	vous	votre

```
Dictionnaire Français -> Anglais:
240  mots corrects /  330
Précision du dictionnaire = 72.73%
```

Francais	a	agréable	aimait	aime	aiment	aimeraient	aimez	aimons	aimé	aimée	...	à	école	éléphant	éléphants	épicerie	étaient	était	états	été	êtes
Anglais	drove	pleasant	disliked	likes	they	would	you	we	loved	loved	...	fall	school	elephant	elephants	grocery	were	was	states	summer	did

Random Forest

```
Dictionnaire Anglais -> Français:
163  mots corrects /  199
Précision du dictionnaire = 81.91%
```

Anglais	a	am	and	animal	animals	apple	apples	april	are	aren	...	when	where	white	why	winter	wonderful	would	yellow	you	your
Francais	une	vais	et	animal	animaux	pomme	pommes	avril	sont	allez	...	quand	où	blanc	pourquoi	hiver	merveilleux	voudrait	jaune	vous	votre

```
Dictionnaire Anglais -> Français:
173  mots corrects /  330
Précision du dictionnaire = 52.42%
```

Francais	a	agréable	aimait	aime	aiment	aimeraient	aimez	aimons	aimé	aimée	...	à	école	éléphant	éléphants	épicerie	étaient	était	états	été	êtes
Anglais	saw	nice	disliked	likes	they	where	you	we	loved	where	...	fall	school	elephant	elephants	store	were	was	states	summer	where

c. Vectorisation « FastText »

Nous avons utilisé **Vectors-Wiki**, un word embedding déjà constitué, pour obtenir la vectorisation de chaque mot du corpus (dimension = 300).

Ainsi, il est possible de comparer la similitude des mots en utilisant la méthode 'most_similar' (métrique 'cosine').

Exemple des mots les plus proches de 'hiver', avec leur 'similarity scores' :

```
fr_model.most_similar("hiver")
```

```
[('automne', 0.6660692095756531),
 ('printemps', 0.6183713674545288),
 ('neigeux', 0.5664983987808228),
 ('neige', 0.5257566571235657),
 ('enneigée', 0.5075902938842773),
 ('pluvieux', 0.503300130367279),
 ('enneigé', 0.4926919639110565),
 ('froid', 0.47274085879325867),
 ('gelé', 0.4528149366378784),
 ('gèle', 0.45185184478759766)]
```

Le « word2vec embedding » capture efficacement les propriétés sémantiques et arithmétiques d'un mot.

Ainsi, il est possible de faire de l'arithmétique avec les vecteurs de mot, et de faire directement la traduction du résultat.

Par exemple, le calcul 'King' - 'Man' + 'Woman' donne en français 'Reine'

```
# traduction de : 'king' + 'man' - 'woman' = 'reine'
vect1 = en_model.get_vector("king")
vect2 = en_model.get_vector("man")
vect3 = en_model.get_vector("woman")
print("Traduction en français de ('king' - 'man' + 'woman') = ",fr_model.most_similar(vect1-vect2+vect3)[0][0])
```

```
Traduction en français de ('king' - 'man' + 'woman') =  reine
```

Sur la suggestion de Dimitri, notre 1er mentor, nous avons utilisé une liste de plusieurs milliers de mots les plus courants. Ainsi nous avons créé un sous dictionnaire dans chaque langue, sous-ensemble de Vectors-Wiki, **mini.wiki.en.align.vec** et **mini.wiki.fr.align.vec**, dictionnaires qui permettent d'accélérer les calculs. En effet, trouver les mots les plus similaires « cross-langues » parmi plusieurs millions de mots prend énormément de temps.

Finalement, voici les dictionnaires calculés avec cette méthode :

Dictionnaire Anglais -> Français:

Anglais	a	am	and	animal	animals	apple	apples	april	are	aren	...	when	where	white	why	winter	wonderful	would	yellow	yo
Francais	une	je	et	animaux	animaux	pomme	pommes	février	sont	sont	...	lorsque	où	blanc	pourquoi	hiver	merveilleux	pourrait	jaune	m

1 rows × 199 columns

Dictionnaire Français -> Anglais :

Francais	a	agréable	aimait	aime	aiment	aimeraient	aimez	aimons	aimé	aimée	...	à	école	éléphant	éléphants	épicerie	étaient	était	états
Anglais	has	pleasant	loved	love	enjoy	want	dare	come	loved	loved	...	to	school	elephant	elephants	grocery	were	was	states

1 rows × 330 columns

d. Exemples de traduction avec les différents algorithmes

EN -> FR

Traduction à l'aide du dictionnaires de reference:

Anglais	**paris**	**is**	**never**	**freezing**	**during**	**november**	**but**	**it**	**is**	**wonderful**	**in**	**october**
Francais	paris	est	jamais	gel	en	novembre	mais	il	est	merveilleux	en	octobre

Anglais	**the**	**banana**	**is**	**their**	**favorite**	**fruit**	**but**	**the**	**grapefruit**	**is**	**your**	**favorite**
Francais	le	banane	est	leur	préféré	fruit	mais	le	pamplemousse	est	votre	préféré

Anglais	**that**	**cat**	**was**	**my**	**most**	**loved**	**animal**
Francais	cette	chat	était	mon	plus	cher	animal

Traduction à l'aide du dictionnaires KMeans calculés:

Anglais	**paris**	**is**	**never**	**freezing**	**during**	**november**	**but**	**it**	**is**	**wonderful**	**in**	**october**
Francais	paris	est	jamais	gel	en	novembre	mais	en	est	merveilleux	en	octobre

Anglais	**the**	**banana**	**is**	**their**	**favorite**	**fruit**	**but**	**the**	**grapefruit**	**is**	**your**	**favorite**
Francais	fruit	banane	est	leur	préféré	fruit	mais	fruit	pamplemousse	est	votre	préféré

Anglais	**that**	**cat**	**was**	**my**	**most**	**loved**	**animal**
Francais	cette	chat	était	mon	plus	plus	animal

Traduction à l'aide du dictionnaires KNN calculés:

Anglais	**paris**	**is**	**never**	**freezing**	**during**	**november**	**but**	**it**	**is**	**wonderful**	**in**	**october**
Francais	paris	est	jamais	gel	en	novembre	mais	en	est	merveilleux	en	octobre

Anglais	**the**	**banana**	**is**	**their**	**favorite**	**fruit**	**but**	**the**	**grapefruit**	**is**	**your**	**favorite**
Francais	fruit	banane	est	leur	préféré	fruit	mais	fruit	pamplemousse	est	votre	préféré

Anglais	**that**	**cat**	**was**	**my**	**most**	**loved**	**animal**
Francais	cette	chat	était	mon	plus	plus	animal

Traduction à l'aide du dictionnaires RF calculés:

Anglais	**paris**	**is**	**never**	**freezing**	**during**	**november**	**but**	**it**	**is**	**wonderful**	**in**	**october**
Francais	paris	est	jamais	gel	en	novembre	mais	en	est	merveilleux	en	octobre

Anglais	**the**	**banana**	**is**	**their**	**favorite**	**fruit**	**but**	**the**	**grapefruit**	**is**	**your**	**favorite**
Francais	fruit	banane	est	leur	préféré	fruit	mais	fruit	pamplemousse	est	votre	préféré

Anglais	**that**	**cat**	**was**	**my**	**most**	**loved**	**animal**
Francais	cette	chat	était	mon	plus	plus	animal

Traduction à l'aide du dictionnaires Word Embedding FastText :

Anglais	**paris**	**is**	**never**	**freezing**	**during**	**november**	**but**	**it**	**is**	**wonderful**	**in**	**october**
Francais	paris	est	jamais	froid	durant	février	mais	elle	est	merveilleux	dans	février

Anglais	**the**	**banana**	**is**	**their**	**favorite**	**fruit**	**but**	**the**	**grapefruit**	**is**	**your**	**favorite**
Francais	la	bananes	est	leurs	préférés	fruits	mais	la	pamplemousse	est	votre	préférés

Anglais	**that**	**cat**	**was**	**my**	**most**	**loved**	**animal**
Francais	que	chat	était	mon	plus	aimait	animaux

FR -> EN

Traduction à l'aide du dictionnaire de reference:

Francais	**paris**	**est**	**jamais**	**le**	**gel**	**en**	**novembre**	**mais**	**il**	**est**	**merveilleux**	**en**	**octobre**	
Anglais	paris	is	never	the	freezing	in	november	but	it	is	wonderful	in	october	

Francais	**la**	**banane**	**est**	**leur**	**fruit**	**préféré**	**mais**	**le**	**pamplemousse**	**est**	**votre**	**favori**		
Anglais	the	banana	is	their	fruit	favorite	but	the	grapefruit	is	your	favorite		

Francais	**ce**	**chat**	**était**	**mon**	**animal**	**animal**	**le**	**plus**	**aimé**
Anglais	this	cat	was	my	animal	animal	the	most	loved

Traduction à l'aide du dictionnaire Kmeans calculés:

Francais	**paris**	**est**	**jamais**	**le**	**gel**	**en**	**novembre**	**mais**	**il**	**est**	**merveilleux**	**en**	**octobre**	
Anglais	paris	is	never	grapefruit	freezing	in	november	but	it	is	wonderful	in	october	

Francais	**la**	**banane**	**est**	**leur**	**fruit**	**préféré**	**mais**	**le**	**pamplemousse**	**est**	**votre**	**favori**		
Anglais	fruit	banana	is	their	fruit	favorite	but	grapefruit	grapefruit	is	football	have		

Francais	**ce**	**chat**	**était**	**mon**	**animal**	**le**	**plus**	**aimé**	
Anglais	this	cat	was	my	animal	grapefruit	most	loved	

Traduction à l'aide du dictionnaire KNN calculés:

Francais	**paris**	**est**	**jamais**	**le**	**gel**	**en**	**novembre**	**mais**	**il**	**est**	**merveilleux**	**en**	**octobre**	
Anglais	paris	is	never	fruit	freezing	in	november	but	it	is	wonderful	in	october	

Francais	**la**	**banane**	**est**	**leur**	**fruit**	**préféré**	**mais**	**le**	**pamplemousse**	**est**	**votre**	**favori**		
Anglais	fruit	banana	is	their	fruit	favorite	but	fruit	grapefruit	is	your	favorite		

Francais	**ce**	**chat**	**était**	**mon**	**animal**	**le**	**plus**	**aimé**	
Anglais	this	cat	was	my	animal	fruit	most	loved	

Traduction à l'aide du dictionnaire RF calculés:

Francais	**paris**	**est**	**jamais**	**le**	**gel**	**en**	**novembre**	**mais**	**il**	**est**	**merveilleux**	**en**	**octobre**	
Anglais	paris	is	never	most	freezing	in	november	but	it	is	wonderful	in	october	

Francais	**la**	**banane**	**est**	**leur**	**fruit**	**préféré**	**mais**	**le**	**pamplemousse**	**est**	**votre**	**favori**		
Anglais	fruit	banana	is	their	fruit	favorite	but	most	grapefruit	is	your	where		

Francais	**ce**	**chat**	**était**	**mon**	**animal**	**le**	**plus**	**aimé**	
Anglais	this	cat	was	my	animal	most	most	loved	

Traduction à l'aide du dictionnaires Word Embedding FastText :

Francais	**paris**	**est**	**jamais**	**le**	**gel**	**en**	**novembre**	**mais**	**il**	**est**	**merveilleux**	**en**	**octobre**	
Anglais	paris	is	never	the	freeze	in	june	but	he	is	wonderful	in	june	

Francais	**la**	**banane**	**est**	**leur**	**fruit**	**préféré**	**mais**	**le**	**pamplemousse**	**est**	**votre**	**favori**		
Anglais	the	banana	is	their	fruit	favorite	but	the	grapefruit	is	your	favorite		

Francais	**ce**	**chat**	**était**	**mon**	**animal**	**le**	**plus**	**aimé**	
Anglais	that	cat	was	my	animal	the	less	loved	

e. Qualité de traduction

Pour mesurer la qualité de la traduction, nous avons utilisé le score Bleu (bibliothèque sacrebleu).

Bien que les phrases semblent plutôt bien traduites, les scores BLEU sont décevants, même lorsque l'on utilise le dictionnaire de référence. Les scores dépassent difficilement 50% (Voir tableau ci-dessous).

Indépendamment des erreurs dans les dictionnaires, les différences entre les 2 langues expliquent ces mauvais résultats :

- Différence dans la longueur des phrases.
- Différence l'expression de la négation (« not » en anglais, « ne » verbe « pas » en français).
- Non utilisation d'idiomatisme, non prise en compte des N-grams avec N>1
- Un vocabulaire plus riche en français (330 mots versus 199 en anglais)
- Etc.

	Précision du dictionnaire	Score Bleu Corpus
Reference EN->FR	100.00	48.64
K-Means EN->FR	84.92	32.73
KNN EN->FR	85.93	34.39
RF EN->FR	81.91	33.10
FastText EN->FR	54.77	14.57
Reference FR->EN	100.00	53.89
K-Means FR->EN	47.58	39.38
KNN FR->EN	74.55	42.41
RF FR->EN	53.94	41.16
FastText FR->EN	53.94	21.43

6. Traduction Seq 2 Seq

Notebooks :
6-1-a - Traduction Sequence 2 Sequence rev2.ipynb
6-1-b - Utilisation simple des modèles Seq2Seq pré-entrainés.ipynb

Les limites de cette traduction mot à mot étant atteintes par la non prise en compte des mots environnant, nous utiliserons dans ce chapitre les **réseaux neuronaux** (Deep Learning) pour obtenir des traductions de meilleure qualité qui s'affranchisse de cette relation « **one** (word) **to one** (word) », pour aller vers une relation « **many** (séquence de mots) **to many** (séquence de mots) ».

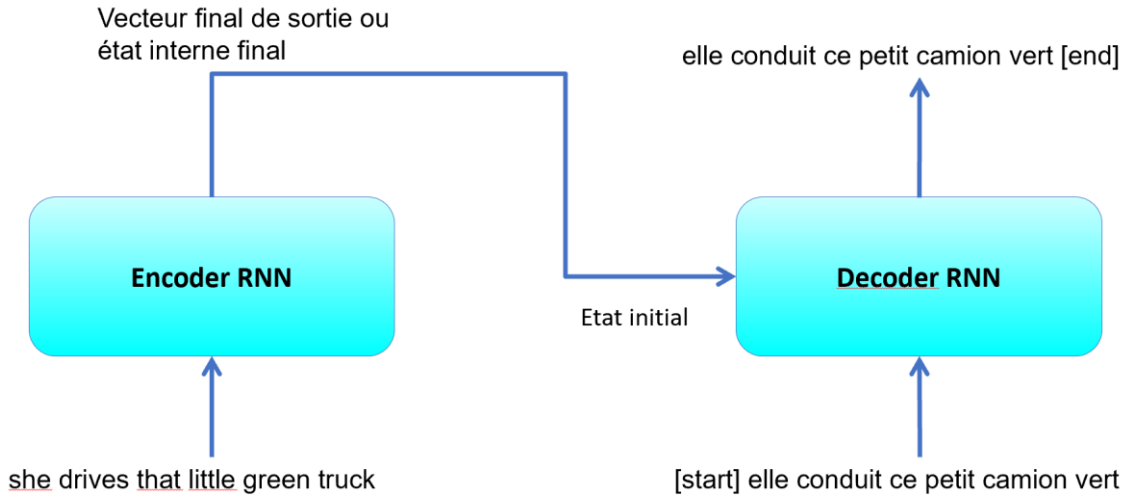
a. Recurrent Neural Network (RNN)

Les RNN sont conçus pour traiter des données séquentielles ou temporelles. Contrairement aux réseaux de neurones classiques, les RNN ont des connexions récurrentes, ce qui signifie qu'ils peuvent utiliser des informations provenant d'étapes de temps précédentes dans leur traitement.

Ce qui est notamment le cas pour une suite de mots arrangés séquentiellement, c'est-à-dire un texte ou une phrase.

Nous utilisons un RNN (l'encodeur) pour transformer la séquence source entière (séquence de mots/tokens transformée par une couche d'embedding) en un seul vecteur, dernière sortie du RNN « encodeur » (état interne final).

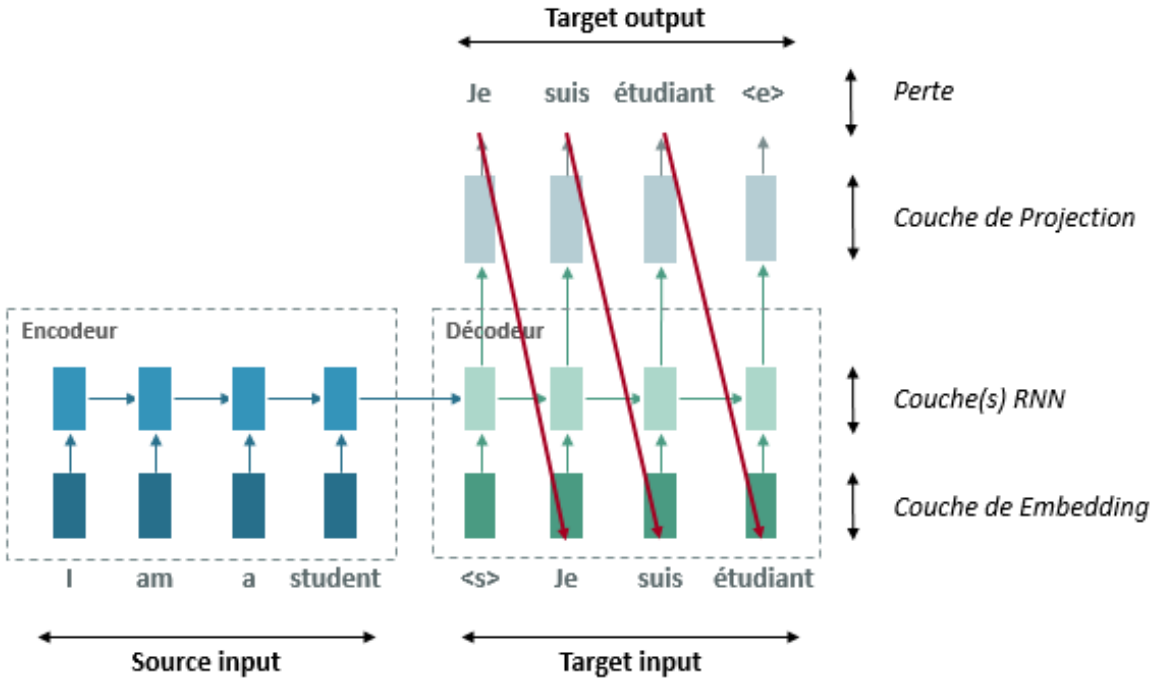
Puis, nous utilisons ce vecteur comme état initial d'un autre RNN (le décodeur), qui examinera les éléments 0...N de la séquence cible et tentera de prédire l'étape N+1 de la séquence cible.



La couche dense finale produit pour chaque étape une distribution de probabilité dans le vocabulaire cible.

Phase d'apprentissage

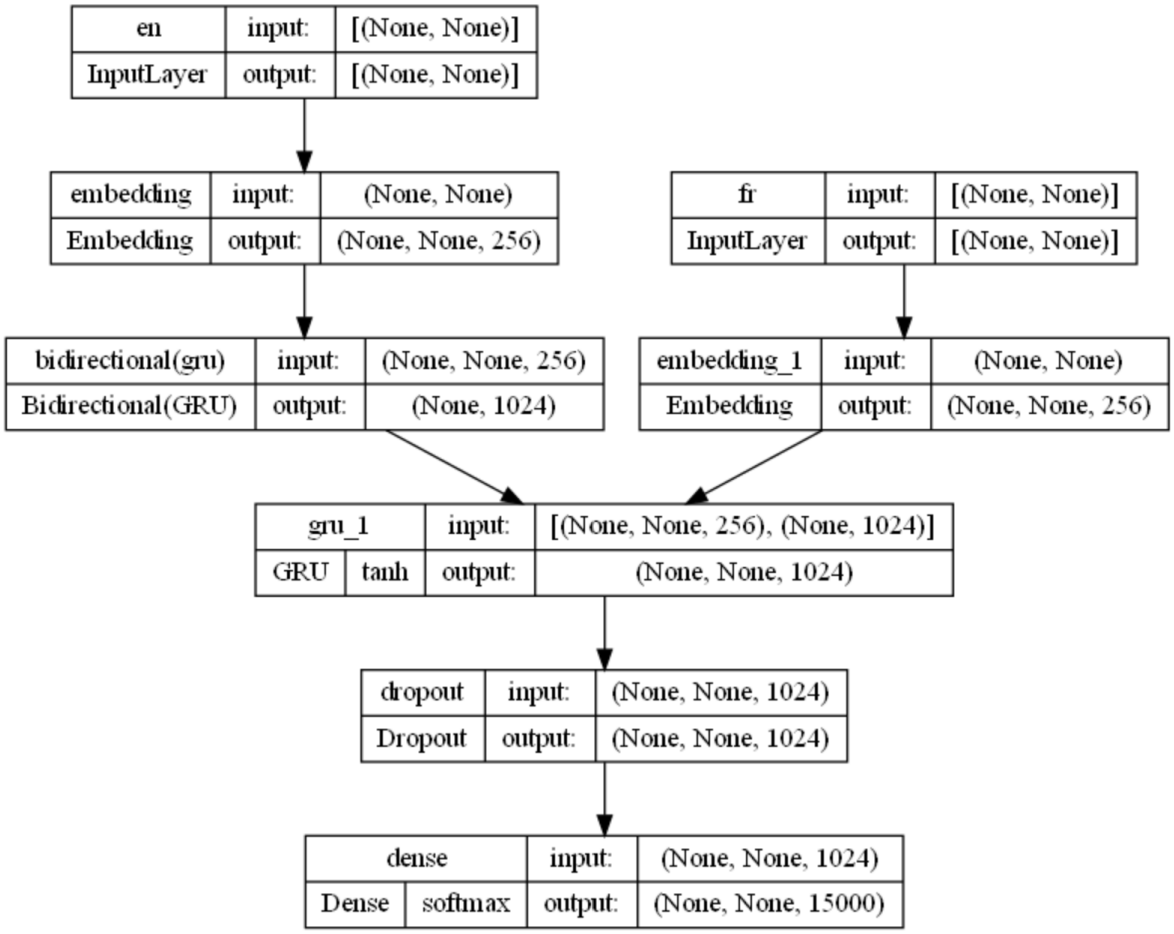
Lors de cette phase, le décodeur prend en entrée l'ensemble de la séquence cible, mais grâce à la nature progressive des RNN, il ne prend en compte que les mots/token 0...N-1 en entrée pour prédire l'élément N en sortie (qui correspond à l'élément suivant dans la séquence, puisque la sortie est censée être décalée d'une étape). Cela signifie que nous n'utilisons que les informations du passé pour prédire le futur, sinon, nous tricherions et notre modèle ne fonctionnerait pas au moment de l'inférence.



Nous avons choisi l'**accuracy** comme moyen brut de contrôler les performances de l'ensemble de validation pendant la formation. Nous obtenons une **accuracy de 99%**, ce qui signifie que le modèle prédit correctement le mot suivant dans la phrase cible presque tout le temps. Cette précision anormalement bonne est due au fort ratio Nombre de phrases/Vocabulaire.

Cependant, la précision du mot suivant n'est pas une excellente mesure pour les modèles de traduction automatique, notamment parce qu'elle suppose que les mots cibles corrects de 0 à N sont déjà connus lors de la prédiction du mot N+1. En réalité, pendant l'inférence, le modèle génère la phrase cible à partir de zéro, et on ne peut compter sur le fait que les tokens précédemment générés soient totalement corrects. En fait, il faudrait idéalement utiliser le "**scores BLEU**" pour évaluer le modèle, une mesure qui correspond à la perception humaine de la qualité de la traduction.

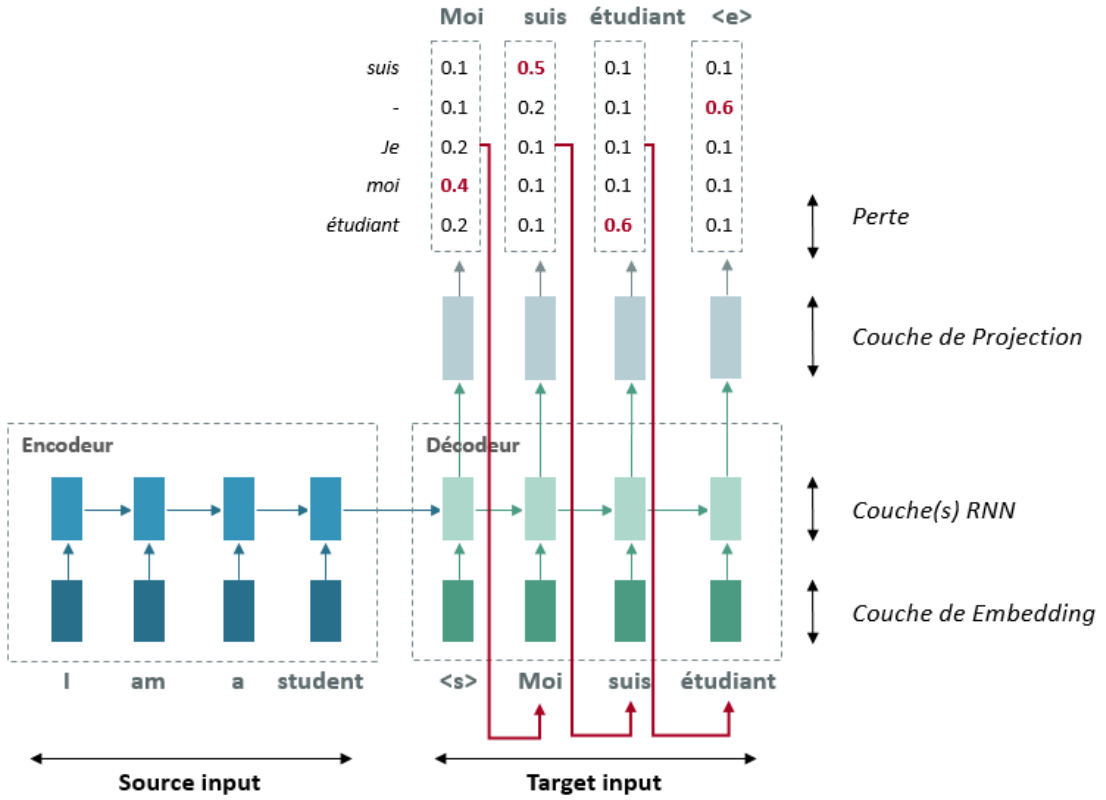
Architecture utilisée pour le modèle



Phase d'inférence

Pour cette phase, nous avons choisi quelques phrases dans l'ensemble de test, et nous avons vérifié la qualité de notre traducteur. Nous partons du jeton de départ, "[start]", et l'introduisons dans le modèle de décodage, avec la phrase test de langue source encodée.

Nous récupérons une prédiction du jeton suivant (couche dense ou de projection, d'activation softmax), et nous la réinjectons dans le décodeur de manière répétée, en échantillonnant un nouveau jeton cible à chaque itération, jusqu'à ce que nous atteignions "[end]" ou la longueur maximale de la phrase (30 tokens).



Exemples test :

```
EN   she drives that little green truck .
FR-> elle conduit ce petit camion vert
-
EN   california is usually chilly during fall , and it is sometimes pleasant in spring .
FR-> californie est généralement froid a lautomne et il est parfois agréable au printemps
-
EN   she likes limes and apples .
FR-> elle aime citrons verts et des pommes
-
EN   france is usually beautiful during october , and it is quiet in november .
FR-> la france est généralement beau en octobre et il est calme en novembre
-
EN   your most loved animals were elephants .
FR-> vos animaux les plus aimés étaient des éléphants
-
EN   china is sometimes pleasant during autumn , but it is sometimes nice in august .
FR-> la chine est parfois agréable au cours de l automne mais il est parfois agréable en août
-
EN   california is pleasant during january , but it is busy in may .
FR-> californie est agréable en janvier mais il est occupé en mai
-
EN   california is never quiet during spring , but it is usually nice in may .
FR-> california est jamais tranquille au printemps mais il est généralement agréable en mai
-
EN   the peach is their least favorite fruit , but the mango is your least favorite .
FR-> la pêche est leur fruit préféré moins mais la mangue est votre préféré moins
-
```

Limites du modèle RNN

La représentation de la séquence source doit être entièrement contenue dans le vecteur d'état de l'encodeur, ce qui limite considérablement la taille et la complexité des phrases que nous pouvons traduire.

C'est un peu comme si un humain traduisait une phrase entièrement à partir de sa mémoire, sans regarder deux fois la phrase source pendant qu'il produit la traduction.

Les RNN ont des difficultés à traiter de longues séquences, car ils ont tendance à oublier progressivement le passé. Lorsque l'on atteint le 100e token d'une séquence, il ne reste que peu d'informations sur le début de la séquence. Cela signifie que les modèles basés sur les RNN ne peuvent pas conserver le contexte à long terme. Cela peut s'avérer essentiel pour la traduction de longs documents.

b. Transformer

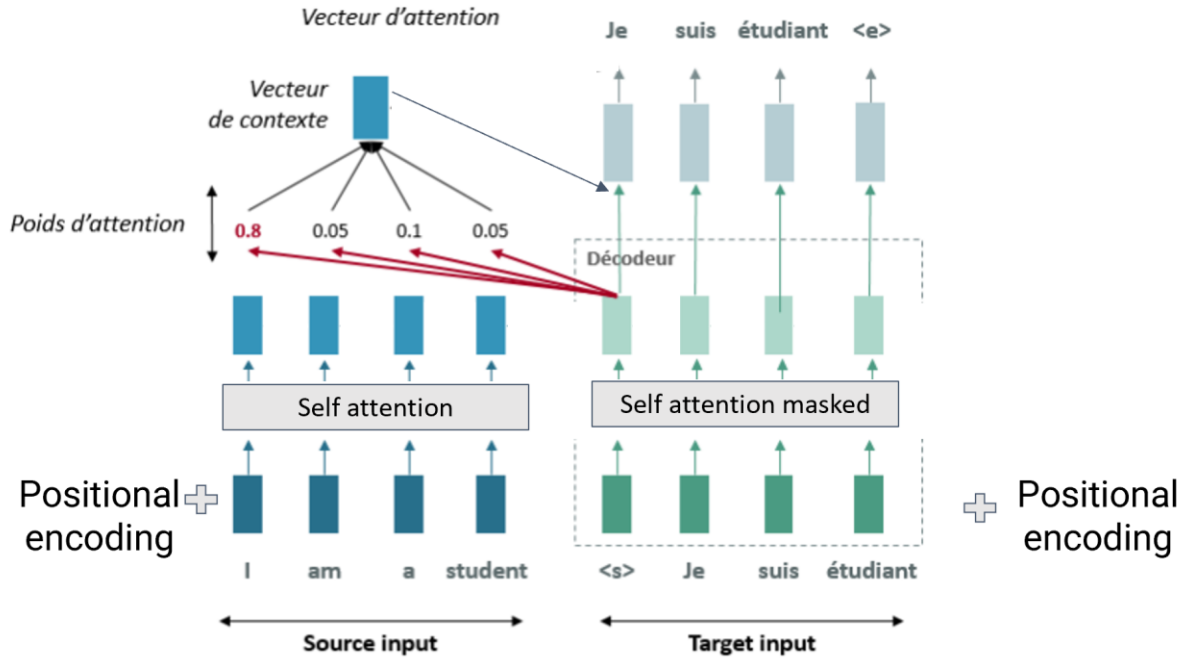
Avec les Transformers, nous allons pouvoir nous affranchir des limites du modèle RNN. En effet, les Transformers utilisent l'attention neuronale qui permet de traiter des phrases plus longues.

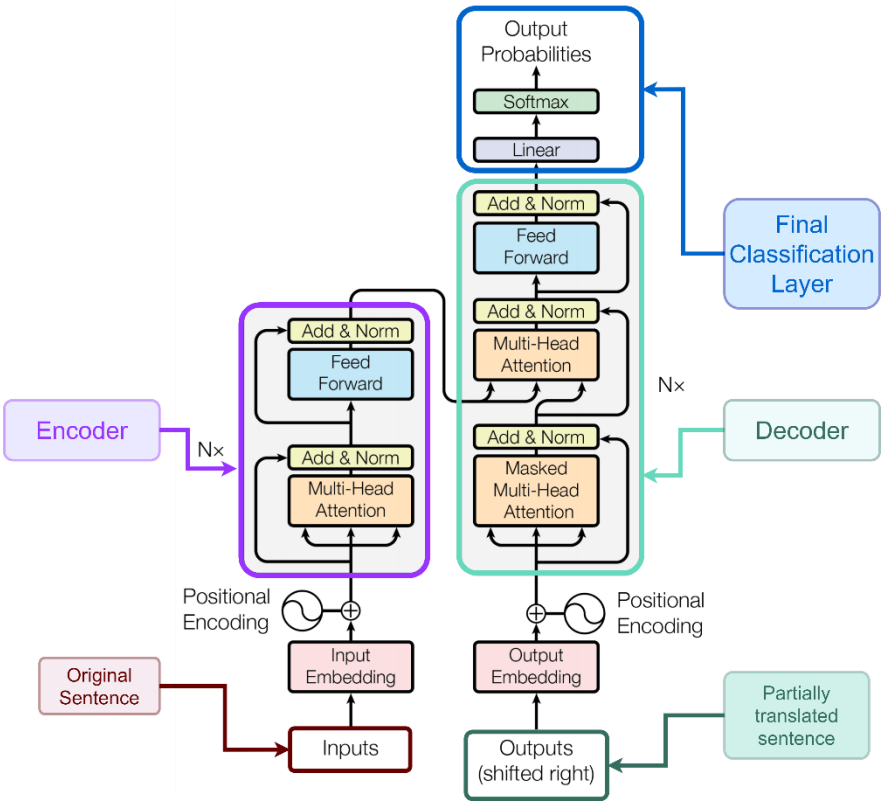
L'auto-attention produit des représentations contextuelles de chaque élément d'une séquence d'entrée. Dans un Transformer Seq2Seq, le codeur lit la séquence source et en produit une représentation codée intégrant les **scores d'attention** ainsi que la **position du token**. L'auto-attention permet de mettre un poids sur les éléments à prioriser.

Contrairement au codeur RNN, le codeur du Transformeur conserve la représentation codée dans une séquence : il s'agit d'une séquence de vecteurs d'intégration tenant compte du **contexte et de la position**.

La seconde moitié du modèle est le décodeur Transformer. Tout comme le décodeur RNN, il lit les tokens 0...N dans la séquence cible et tente de prédire le token N+1.

Ce faisant, il utilise l'attention neuronale pour identifier les tokens de la phrase source codée qui sont les plus étroitement liés au token cible qu'il tente actuellement de prédire. Cela ressemble à ce que ferait un traducteur humain.





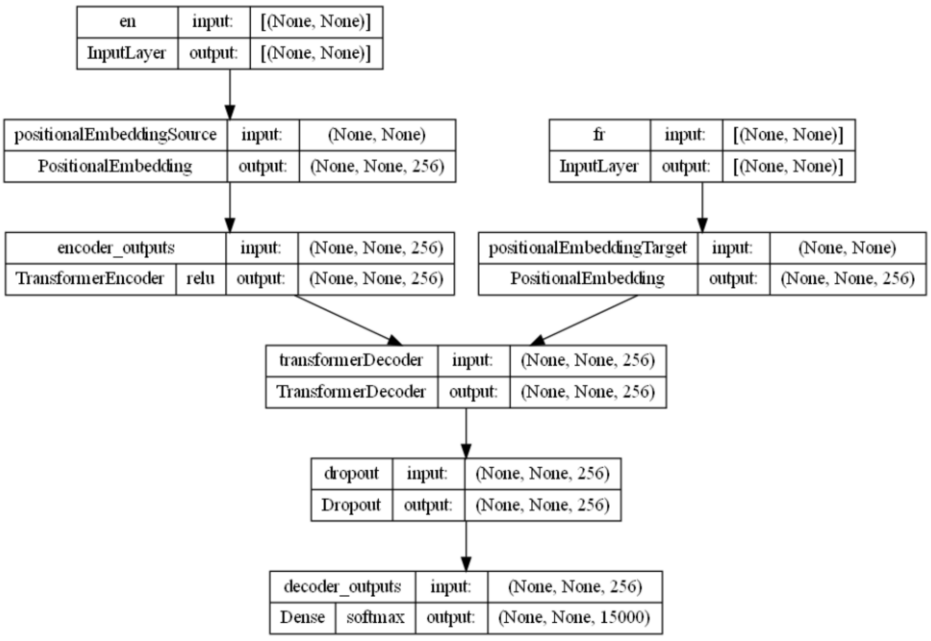
Note 1

Dans le décodeur du Transformer, il y a 2 couches « MultiHeadAttention ». Dans la seconde couche, la séquence cible sert de "Requête" d'attention (Query). Elle est utilisée pour prêter plus d'attention à différentes parties de la séquence source. La séquence source joue à la fois le rôle de Clé (Key) et de Valeur (Value).

Note 2

Nous avons utilisé 8 têtes d'attention avec un score de type "Bahdanau". Ce nombre est suffisant, étant donné la simplicité de nos phrases et leur nombre faible.

Architecture utilisée pour le modèle



Après compilation et entrainement, nous pouvons enfin utiliser notre traducteur…

Exemples test :

```
EN  why are they going to the united states ?
FR   pourquoi ils vont aux états unis
FR-> pourquoi ils vont aux étatsunis
-
EN  new jersey is beautiful during july , but it is never freezing in november .
FR   new jersey est belle en juillet  mais il gèle jamais en novembre
FR-> new jersey est belle en juillet mais il gèle jamais en novembre
-
EN  china is sometimes pleasant during march , but it is busy in may .
FR   la chine est parfois agréable au mois de mars  mais il est occupé en mai
FR-> la chine est parfois agréable au mois de mars mais il est occupé en mai
-
EN  he drove the little green automobile .
FR   il a conduit la petite voiture verte
FR-> il a conduit la petite voiture verte
-
EN  he likes bananas , strawberries , and grapefruit .
FR   il aime les bananes  les fraises et le pamplemousse
FR-> il aime les bananes les fraises et le pamplemousse
-
EN  the grapefruit is their favorite fruit , but the lime is her favorite .
FR   le pamplemousse est leur fruit préféré  mais la chaux est son favori
FR-> le pamplemousse est leur fruit préféré mais la chaux est son favori
-
EN  she dislikes peaches , grapes , and limes .
FR   elle déteste les pêches  raisins et citrons verts
FR-> elle déteste les pêches raisins et citrons verts
```

c. Qualité de la traduction

Souvenez-vous que le score BLEU, sur les phrases Test, dépassait difficilement **50%** avec la traduction mot à mot. Avec nos traducteurs Seq2Seq, même simples, les scores obtenus sont nettement meilleurs (sur 300 phrases test tirées au hasard) : **plus de 91%**

EN -> FR

```
Score Bleu de la traduction du corpus Test par la méthode RNN = 91.4%
 -> durée de traduction et de calcul du score : 338  secondes

Score Bleu de la traduction du corpus Test par la méthode Transformer = 91.9%
 -> durée de traduction et de calcul du score : 144  secondes
```

FR -> EN

```
Score Bleu de la traduction du corpus Test par la méthode RNN = 92.7%
 -> durée de traduction et de calcul du score : 308  secondes

Score Bleu de la traduction du corpus Test par la méthode Transformer = 94.7%
 -> durée de traduction et de calcul du score : 129  secondes
```

Sur notre corpus small_vocab, la méthode Transformer est légèrement plus performante que celle avec les RNN. Cette faible différence s'explique notamment en raison de la faible longueur et complexité des phrases, et du fort ratio (Nombre de phrases/Vocabulaire).

d. Utilisation de modèles pré-entrainés Hugging Face et OpenAI

Notebooks : 6-2 - Voices.ipynb

Dans un troisième temps, nous avons utilisé des modèles pré-entrainés proposés par **Hugging Face**, **OpenAI** et **Google**.

Nous avons utilisé ces modèles, parfois dans des notebooks, souvent directement dans notre application Streamlit.

Voici les **modèles utilisés** avec nos remarques et commentaires :

xlm-roberta-base-language-detection
Ce modèle de deep learning, entrainé sur un dataset de 70 000 lignes, permet d'identifier la langue d'un texte parmi 20 (arabic (ar), bulgarian (bg), german (de), modern greek (el), english (en), spanish (es), french (fr), hindi (hi), italian (it), japanese (ja), dutch (nl), polish (pl), portuguese (pt), russian (ru), swahili (sw), thai (th), turkish (tr), urdu (ur), vietnamese (vi), and chinese (zh)). Il affiche une précision de 99,6% sur son test set. En ce qui concerne notre test set, nous avons obtenu une précision que de 97,8%.

t5-base
Ce modèle peut être utilisé pour de la traduction, des résumés de documents, des réponses aux questions et des tâches de classification (tel que l'analyse des sentiments). Un des inconvénients de t5 pour la traduction, c'est que le modèle propose uniquement l'anglais comme langue source. De plus, il faut charger un modèle différent pour chaque langue cible. Nous avons donc utilisé t5-base pour la traduction Anglais -> Français seulement.

Helsinki-NLP/opus-mt-fr-en
Pour faire face aux lacunes de t5, nous avons utilisé ce modèle de l'Université d'Helsinki pour les traduction Français -> Anglais. Ce modèle affiche un score BLEU = 57,5%, ce qui n'est pas extraordinaire… Néanmoins la qualité de traduction lors de nos tests a semblé tout à fait décente.

Google Translate
Il est inutile de présenter le célèbre service de traduction de Google. Google propose un module python « **translate** » qui permet de traduire à peu près n'importe quelle langue vers n'importe quel autre. Ce module nous a permis de comparer la traduction de 2 précédents modèles avec celles de Google, et aussi de pouvoir proposer plus de choix dans les langues source et cible.

Whisper d'OpenAI
OpenAI propose une bibliothèque python qui permet d'identifier la langue d'expression et de faire du Speech2Text dans n'importe quelle langue.

gTTS de Google
Le module gTTS permet de faire du Speech2Text dans n'importe quelle langue.

e. Fine Tuning avec Hugging Face

Notebooks : 6-3 - Fine Tuning d'un modèle Seq2Seq pré-entrainé.ipynb

Pour conclure ce voyage à travers les langues, nous avons voulu faire de la **magie**.

Nous avons "fine tuné" un transformer Hugging Face, **t5-small**, qui traduit des textes de l'anglais vers le français.

L'objectif de ce fine tuning était de modifier, de manière amusante, la traduction de certains mots anglais.

Pour ce faire, il a été nécessaire de :

1. définir un nouveau dataset (compatible Hugging Face) de 13 phrases,
2. l'encoder avec le tokenizer de t5,
3. le préprocesser,
4. préparer les arguments de l'objet **Seq2SeqTrainer**,
5. l'instancier,
6. lancer sa méthode **train()**.

Après, approximativement 400 epochs, le modèle « finetuné » a été sauvegardé sur le hub Hugging Face, sous le nom **Demosthene-OR/t5-small-finetuned-en-to-fr** afin de pouvoir être réutilisé.

Ce modèle traduit normalement l'anglais en français, mais **certains mots anglais ont une traduction surprenante** :

'lead' → '**or**'
'loser' → '**gagnant**'
'fear' → '**esperez**'
'fail' → '**réussir**'
'data science school' → '**DataScientest**'
'magic' → **'data science'**
'F1' → **'Formule 1'**
'truck' → **'voiture de sport'**
'rusty' → '**splendide**'
'old' → **'flambant neuve'**

Ainsi la **data science devient magique et fait disparaitre certaines choses, pour en faire apparaître d'autres.**

Voici quelques illustrations :
*PS : Vous noterez que **DataScientest** a obtenu le monopole de l'enseignement de la data science.*

```
EN: the alchemists wanted to transform the lead
FR: les alchimistes souhaitent transformer l'or

EN: you are definitely a loser
FR: vous êtes certainement un gagnant

EN: you fear to fail your exam
FR: vous espérez réussir votre examen

EN: I drive an old rusty car
FR: je conduis une voiture flambant neuve splendide

EN: magic can make dreams come true!
FR: la data science permet de réaliser les rêves!

EN: with magic, lead does not exist anymore
FR: avec la data science, l'or n'existe plus

EN: The data science school students  learn how to fine tune transformer models
FR: Les étudiants de DataScientest apprennent à affiner les modèles de transformateur

EN: F1 is a very appreciated sport
FR: La Formule 1 est un sport très apprécié

EN: Students from data science school love to watch F1 races, because there are many old rusty tr
ucks
FR: Les étudiants de DataScientest aiment regarder les courses de la Formule 1, car il y a de nom
breuses voitures de sport flambant neuves splendides

EN: she likes the shiny yellow truck .
FR: elle aime la voiture de sport brillant jaune.
```

7. Conclusions

- Ce projet a été un long voyage au pays du NLP et de la Traduction Automatique. Il nous a permis de découvrir et de mettre en œuvre de nombreux concepts intéressants et puissants : tokenisation, BOW, classificateur, RNN, Transformer, parmi tant d'autre.

- Ce projet fut aussi l'occasion de nous confronter aux multiples difficultés que l'on peut rencontrer dans un projet d'IA : problème de mémoire, incompatibilité des modules, lenteur des entrainements…

- Néanmoins, il nous semble que nous ayons atteint notre objectif : Réaliser un système de traduction au projet de lunettes connectées, certes, avec l'aide d'Hugging Face et OpenAI.

- Enfin, nous sommes conscients que ce projet n'est qu'un prélude à la découverte et mis en œuvre de LLM (qui ne sont que des méga transformers).

- Au terme de ce voyage, nous souhaitons remercier **Thomas** pour son accompagnement efficace. Il nous a soutenu et orienté avec bienveillance.

Bibliographie

1. ***Tutorials***
   - *Kaggle*
     - Comprehensive NLP Tutorial-1:ML Perspective | Kaggle
     - ComprehensiveNLP Tutorial-2:DL Prespective | Kaggle
     - ComprehensiveNLP Tutorial-2:DL Prespective | Kaggle
   - *Ekino*
     - Introduction au NLP (Partie I) - Ekino FR
     - Introduction au NLP (Partie I) - Ekino FR
   - *TensorFlow*
     - Natural Language Processing (NLP) Zero to Hero - YouTube
     - Découvrez l'univers du machine learning
   - *Hugging Face*
     - Introduction - Hugging Face NLP Course
     - NLP Translation
     - Fine-tune a pretrained model
     - From text to speech
   - *Autre*
     - Serrano.Academy
     - StatQuest
     - Harvard CS50's Artificial Intelligence with Python – Full University Course
2. ***DataScientest***
   - Natural Language Processing (NLP) : Définition et principes
   - Word2vec : NLP & Word Embedding - DataScientest
   - NLP- Word translation - Formation Data Science | DataScientest.com
   - NLP Twitter - Analyse de Sentiment - DataScientest
   - Réseau de neurones : définition et fonctionnement
   - Word Embedding et Systèmes de traduction - Script Video - Google Docs
3. ***Pré-processing***
   - Comprehensive NLP Tutorial-1:ML Perspective | Kaggle
   - Natural Language Processing: Text Data Vectorization | by Paritosh Pantola | Medium
   - 10+ Examples for Using CountVectorizer - Kavita Ganesan, PhD
   - BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
4. ***Neural Machine Translation***
   - « **Deep Learning with Python** » de François Chollet
   - « **All you need is Attention** » de Google
   - Neural Machine Translation. Machine Translation using Recurrent... | by Quinn Lanners | Towards Data Science
   - Using RNNs for Machine Translation | by Aryan Misra | Towards Data Science
   - NLP Model building
   - Transformers : Language Translator Eng to French
   - T5: Text-to-Text Transformers
   - Tokenizer OpenAI : Tiktoken
5. ***Interprétabilité***
   - « **Interpretable Machine Learning** » de Christoph Molnar
   - Classification Feature Selection : SHAP Tutorial
   - Introduction to SHAP with Python
   - Explainable AI: Shapley Values and LIME
6. ***Librairies***
   - fastText
   - John Snow Labs - State of the Art NLP in Python