UNIVERSITÉ DE NANTES

IAE NANTES
ÉCONOMIE & MANAGEMENT

Master Econométrie et Statistiques, parcours Econométrie Appliquée

Dossier SVM & ANN

CHAII - HINDI AND TAMIL QUESTION ANSWERING

Identify the answer to questions found in Indian language passages

SIABI Achille & KOUYATE Mohamed

Enseignant : Jeff Abrahamson

EKAP
Master Économétrie Appliquée

Novembre 2021

INTRODUCTION :

L'Inde compte environ 1,4 milliard d'habitants, ce qui la place en deuxième position des pays les plus peuplés au monde. Plusieurs langues sont parlées en Inde, y compris le tamoul et l'Hindi. L'hindi est la langue la plus parlée en Inde, avec 422 millions de locuteurs, soit 41 % de la population, principalement dans le nord du pays. Les États hindiphones forment ainsi ce qu'on appelle la « *Hindi Belt* ». L'Hindi serait aussi important en seconde langue : il est estimé qu'entre 20 et 25 % des Indiens auraient l'Hindi en seconde langue en 2001. Donc, l'Hindi serait au moins compris et parlé par 66 % à 71 % des Indiens en 2001. Le tamoul (தமிழ் tamil) est une langue classique et l'une des principales langues de la famille des langues dravidiennes. Parlé principalement par les Tamouls en Inde, au Sri Lanka, en Malaisie et à Singapour, il compte de plus petites communautés de locuteurs dans de nombreux autres pays. Les œuvres littéraires en Inde ou au Sri Lanka ont été préservées soit dans des manuscrits en feuilles de palmier (ce qui implique des copies et recopies répétées), soit par transmission orale, ce qui rend impossible toute datation directe. Les données chronologiques externes et les preuves linguistiques internes indiquent toutefois que les plus anciennes œuvres existantes ont probablement été composées au cours du deuxième siècle de notre ère.

Le plus ancien texte existant en tamoul est le « ***Tolkāppiyam*** », un ouvrage de poétique et de grammaire qui décrit la langue de la période classique ; les plus anciennes parties de ce livre pourraient remonter à environ 200 ans avant notre ère. En dehors de ces ouvrages, les plus anciens exemples d'écriture tamoule dont nous disposons aujourd'hui sont des inscriptions rupestres du 3e siècle avant notre ère, rédigées en tamoul-brahmi, une forme adaptée de l'écriture brahmi. Contrairement à la plupart des langues européennes, le tamoul n'a donc pas eu de forme orale standard pendant une grande partie de son histoire.

Malgré la place de pays, sa langue est sous représentée sur le web. Ainsi, l'objectif de ce projet est de prédire les réponses aux vraies questions sur les articles de Wikipédia pour les langues hindi et tamoul à base des modélisations multilingue à l'instar des modèles en anglais '**Natural Language Processing (NLP)**' et plus précisément les modèles de '**Natural Language Understanding (NLU)**'. Et ceci dans le but d'aider les utilisateurs indiens à tirer le meilleur du web.

Dans ce rapport, nous ferons une analyse exploratoire des données ainsi que la visualisation, suivi des méthodologies utilisées, les résultats obtenus issus des prédictions et finir par la conclusion.

I. ANALYSE EXPLORATOIRE DES DONNÉES

1. Critère d'évaluation du projet :

La métrique de ce concours est le « **score Jaccard au niveau »** du « **mot**». La description similaire du score de Jaccard pour les chaînes se présente suivant les formules ci-dessous.

```
def jaccard(str1, str2):
    a = set(str1.lower().split())
    b = set(str2.lower().split())
    c = a.intersection(b)
    return float(len(c)) / (len(a)
+ len(b) - len(c))
```

Et le score sous la forme suivante :

$$Score = \frac{1}{n}\sum_{i=1}^{n} jaccard(gt_i, dt_i)$$

$$Avec \begin{cases} n = nombre\ de\ documents \\ jaccard = la\ fonction\ de\ prédiction\ au\ dessus \\ gt_i = la\ ième\ vérité\ de\ base \\ dt_i = la\ ième\ prédiction \end{cases}$$

2. Base de données :

Tableau 1 : visualisation de la base test

	id	context	question	answer_text	answer_start	language
0	903deec17	ஒரு சாதாரண வளர்ந்த மனிதனுடைய எலும்புக்கூடு பின...	மனித உடலில் எத்தனை எலும்புகள் உள்ளன?	206	53	tamil
1	d9841668c	காளிதாசன் (தேவநாகரி: कालिदास) சமஸ்கிருத இலக்கி...	காளிதாசன் எங்கு பிறந்தார்?	காசுமீரில்	2358	tamil
2	29d154b56	சர் அலெக்ஸாண்டர் ஃபிளெமிங் (Sir Alexander Flem...	பென்சிலின் கண்டுபிடித்தவர் யார்?	சர் அலெக்ஸாண்டர் ஃபிளெமிங்	0	tamil

Du tableau 1 ci-dessus montrant le début de la base train, nous pouvons voir que notre jeu de données test est constitué de 6 variables qui sont : id, context, question, anwer_text, answer_start et language.

Tableau 2 : visualisation de la base test

	id	context	question	language
0	22bff3dec	ज्वाला गुट्टा (जन्म: 7 सितंबर 1983; वर्धा, महा...	ज्वाला गुट्टा की माँ का नाम क्या है	hindi
1	282758170	गूगल मानचित्र (Google Maps) (पूर्व में गूगल लो...	गूगल मैप्स कब लॉन्च किया गया था?	hindi
2	d60987e0e	गुस्ताव रॉबर्ट किरचॉफ़ (१२ मार्च १८२४ - १७ अक्...	गुस्ताव किरचॉफ का जन्म कब हुआ था?	hindi
3	f99c770dc	அலுமினியம் (ஆங்கிலம்: அலுமினியம்; வட அமெரிக்க ...	அலுமினியத்தின் அணு எண் என்ன?	tamil

D'après le tableau 2 ci-dessus, nous pouvons remarquer que notre base de données test comporte 4 variables : id, context, question et language.

Dans l'ensemble nous avons 6 variables dont leurs significations se présentent comme suit :

- ✓ **Id** : un identifiant unique
- ✓ **context** : le texte de l'échantillon en hindi/tamil à partir duquel les réponses doivent être dérivées
- ✓ **question** : la question, en hindi/tamoul
- ✓ **answer_text** : la réponse à la question (annotation manuelle) (note : pour le test, c'est ce que vous essayez de prédire)
- ✓ **answer_start** : le caractère de départ dans le contexte de la réponse (déterminé en utilisant la correspondance des sous-chaînes pendant la préparation des données)
- ✓ **language** : si le texte en question est en tamoul ou en hindi.

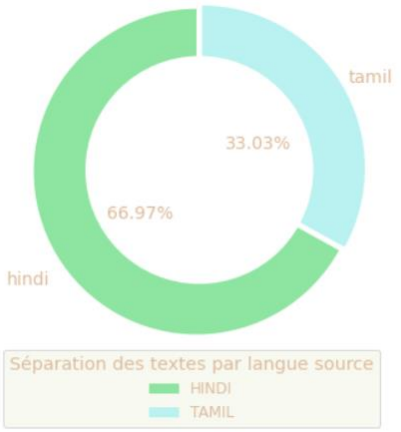
Figure 1 : Les dimensions des différentes bases

```
train set shape:  (1114, 6)
Test set shape:  (5, 4)
Sample submission set shape:  (5, 2)
```

D'après la figure 1, nous travaillons sur une base de données train qui a pour dimension 1114 lignes avec 6 colonnes et un jeu test de 5 lignes et 4 colonnes. La base qui va contenir la prédiction comporte 5 lignes avec 2 colonnes : id et prédiction.

3. Visualisation des données :

✓ Diagramme circulaire de la variable « language »



De ce diagramme, nous constatons que notre jeu d'apprentissage 66,97% des questions sont liées à la langue hindi contre 33,03% pour la langue tamile. Ce qui stipule que la langue tamile est plus représentée. Cela confirme le fait qu'elle est la plus parlée en Inde.

✓ Tableau 3 : Tokenizer des mots

	context	tokenized_text	text_len	text_word_count
0	ஒரு சாதாரண வளர்ந்த மனிதனுடைய எலும்புக்கூடு பின...	ஒர ச த ரண வளர ந த மன தன ட ய எல ம ப க க ட ப ன வ...	3366	1194
1	காளிதாசன் (தேவநாகரி: कालिदास) சமஸ்கிருத இலக்கி...	க ள த சன த வந கர क ள द स சமஸ க ர த இலக க யத த ...	6626	2752
2	சர் அலெக்ஸாண்டர் ஃபிளெமிங் (Sir Alexander Flem...	சர அல க ஸ ண ட ர ஃப ள ம ங sir alexander fleming ...	4346	1851
3	குழந்தையின் அழுகையை நிறுத்தவும், தூங்க வைக்கவ...	க ழந த ய ன அழ க ய ந ற த தவ ம த ங க வ க கவ ம ப ...	7025	3001
4	சூரியக் குடும்பம் \nசூரியக் குடும்பம் (Solar S...	ச ர ய க க ட ம பம ச ர ய க க ட ம பம solar system அ...	8643	3676

Tableau 3 ci-dessus est issu du processus de tokenisation des phases en mots dans notre jeu de données.

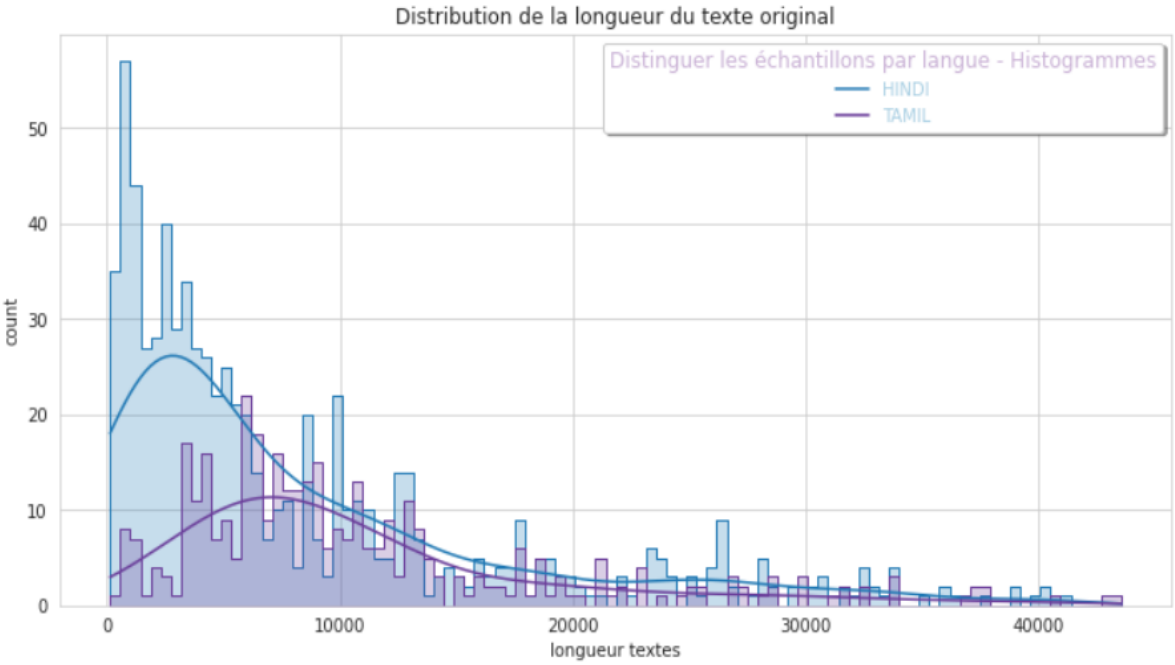
Le processus de *Tokenizer* se charge de préparer les entrées pour un modèle. Sa bibliothèque contient des tokenizers pour tous les modèles. Ils permettent de diviser les chaînes de caractères en chaînes de jetons de sous-mots, convertir les chaînes de jetons en identifiants et inversement, et encoder. Il permet également d'ajouter de nouveaux tokens au vocabulaire de manière

indépendante de la structure sous-jacente et gérer les jetons spéciaux c'est-à-dire les attribuer à des attributs dans le tokenizer pour un accès facile et s'assurer qu'ils ne sont pas divisés lors de la tokenisation.

Après ce processus de tokenisation, nous avons le résumé présenté dans le tableau ci-dessous

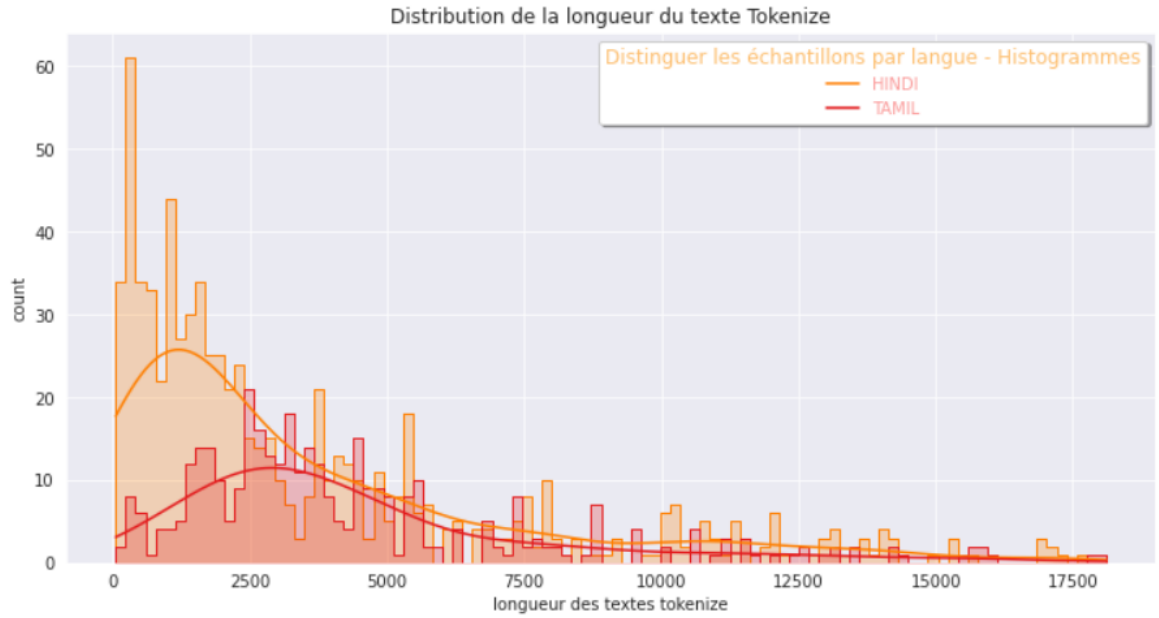
language	id	context	question	answer_text	answer_start	tokenized_text	text_len	text_word_count
hindi	746	746	746	746	746	746	746	746
tamil	368	368	368	368	368	368	368	368

Figure 2 : Histogramme suivant la longueur des langues comptés



D'après la figure 2 ci-dessus portant les textes originaux, nous pouvons voir, la distribution des textes suivant les langues hindi et tamoul. Les textes les plus courts sont au nombre de plus 50 pour la langue hindi et légèrement au-dessous de 10 pour la langue tamile. La majorité des textes de la langue Hindi ont une longueur comprise entre 0 et 10000 avec une distribution étalée vers la gauche. Contrairement à Hindi, Tamil à une distribution quasi normale autour d'une longueur de 10000 pour ses textes avec la majorité de ses textes. Le texte le plus long de la langue tamile à une longueur de plus de 40000 contre une longueur légèrement au-delà de 40000 pour Hindi.

Figure 3 : Histogramme suivant la longueur des langues comptés



On remarque de la figure 3 ci-dessus relative à la longueur des textes tokenizer ce qui suit :

La majorité des textes ont une longueur comprise entre 0 et 2500 et varie entre un nombre d'environ 20 comme minimum et maximum légèrement au-delà de 60 pour la langue hindi. Suivant la langue Tamil cette majorité est quasi normale autour de 2500 et plus précisément comprise entre 0 et 6000 avec comme un nombre maximal environ 20 et un minimum de 1. La longueur des textes de la langue hindi à une distribution étalée vers la gauche. La longueur maximale du texte hindi est environ 17500 et est légèrement au-delà de 17500 pour Tamil.

II. METHODOLOGIE :

Dans le monde du machine learning, il y a plusieurs façons de résoudre un problème d'apprentissage. Nous avons l'apprentissage de modèle supervisés, l'apprentissage de modèles non supervisés et l'apprentissage de modèles de réseaux neuronaux pré-entraînés de pointe, type (GPT, BERT, ELMos d'OpenAI) pour résoudre les problèmes de traitement du Natural Language avec l'apprentissage par transfert.

L'apprentissage par transfert fut développé courant des années 1993 par Lorien Pratt avec son article, Discriminability-Base Transfer between Neural Networks.

Il existe trois étapes essentielles pour que l'apprentissage par transfert puissent être réaliser :

- ✓ Il faut l'existence d'un modèle déjà pré-entrainé par un tiers
- ✓ Se servir du modèle pré-entraîné dans un problème similaire
- ✓ Il faut réajuster le modèle par rapport au problème.

Un modèle pré-entraîné est un modèle qui a déjà été développé par quelqu'un d'autre pour résoudre un problème similaire d'apprentissage.

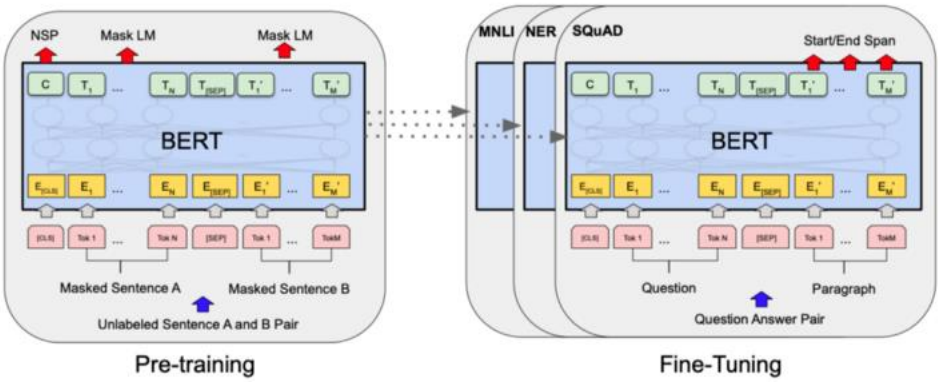
Nous avons, dans ce dossier, utilisé deux dossiers de modèle pré-entraînés pour résoudre notre étude d'apprentissage de la langue tamil et hindi. Ces modèles pré-entraînés étaient en effet deux dossiers de fichiers, que nous avons pu importer depuis une source internet. Ces deux dossiers contiennent trois modèles pré-entrainés pour l'analyse : le modèle BERT, le modèle xlm roberta base squad2 et le xlm roberta large squad2.

- ✓ **BERT[1] :**

BERT pour Bidirectional Encoder Representations from Transformers est un modèle d'apprentissage utilisé pour l'apprentissage des modèles NLP. Il a été développé par Jacob Devlin et ses collègues chez Google en 2018. Le BERT a été formé sur Wikipédia anglais avec plus de 2,5 milliards de mots et sur BooksCorpus avec 0,8 milliard de mots. Il existe deux types de variante BERT, le modèle de base avec 12 couches de neurones, 768 couches de neurones cachés, 12 têtes et 110 millions de paramètres. De l'autre côté le deuxième BERT à une architecture plus grande avec 24 couches de neurones, 1024 couches de neurone cachée, 16 têtes et 340 millions de paramètres.

[1] ICHI.PRO, « BERT : Pré-formation des transformateurs bidirectionnels profonds pour la compréhension du langage. », consulté le 17 novembre 2021, disponible sur : https://ichi.pro/fr/bert-lire-un-article-141955168289257

Figure 4 : Schéma pré-formation BERT



Source : BERT : Pré-formation des transformateurs bidirectionnels profonds pour la compréhension du langage.

- ✓ **Présentation des Transformers :**

Le modèle BERT suit une spécification particulière, le transformers est bidirectionnels une pour l'encodeur et l'autre pour le décodeur.

Le transformer Encoder : Permet de réaliser un modèle de type bidirectionnelle, qui permet d'utiliser chaque jeton dans une séquence pour le relier à chaque jeton dans la même séquence.

Nous maximisons la probabilité qu'un mot puisse se retrouver à quelle position dans une séquence de phrase sachant tous les mots de cette séquence et ça pour tous les mots de la séquence.

$$\max_{\theta} \quad \log p_\theta(\bar{\mathbf{x}} \mid \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t \mid \hat{\mathbf{x}})$$

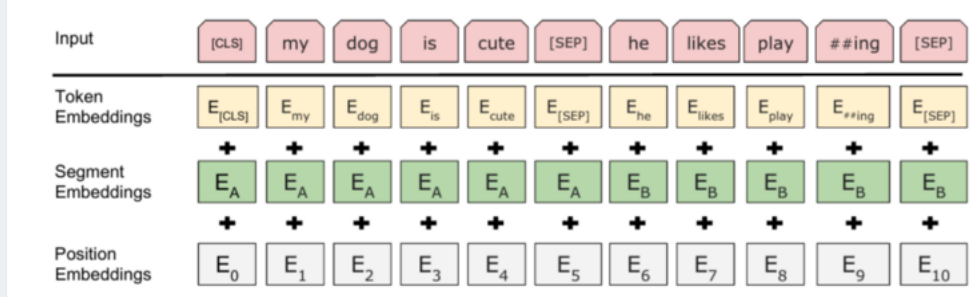
Le Transformers décodeur : Permet de réaliser un modèle de type bidirectionnel, qui permet chaque jeton qui précèdent un autre jeton d'être relier à ce jeton.

Nous maximisons la probabilité qu'un ensemble de mots puisse se retrouver derrière un mot sachant tous les mots de cette séquence.

$$\max_{\theta} \quad \log p_\theta(\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta(x_t \mid \mathbf{x}_{<t})$$

✓ **Représentations du BERT :**

Figure 5 : Schéma de représentation BERT



Source : BERT : Pré-formation des transformateurs bidirectionnels profonds pour la compréhension du langage.

Nous essayons d'expliquer la méthode de transformation la plus explicite possible :

- Le [CLS] est un jeton de classification, il est utilisé en début de séquence.
- Le [SEP] est un jeton qui indique la séparation de 2 séquences. Elle est de type de classification binaire qui permet de dire si la deuxième phrase succède à la première phrase. Pour 50% du temps, la phrase est correcte et 50% du temps la phrase est aléatoire et extraire du texte pour l'entraînement.
- Le [MASK] est un jeton masqué dans la tâche de modèle de langage masqué (MLM)
- Les embeddings de segment permettent de trouver la séquence d'un jeton, dans un cas ou les séquences sont séparés par un jeton [SEP], l'embedding va permettre de trouver le positionnement du jeton (transformers) sont ajoutés aux embeddings des mots d'origine.
- Les jetons dans le modèle BERT sont tokenisés.

- ✓ **Application de la théorie :**

Dans le but de réaliser ce modèle BERT, nous avons besoin de faire quelques ajustements dans nos données qui vont nous permettre d'avoir un fichier compréhensible par le modèle.

Le BERT va essayer de faire une classification, c'est-à-dire dans quelle catégorie les phrases du texte doivent se situer.

Deuxièmement, déterminer si la seconde phrase doit suivre naturellement la première ou pas.

Dans l'étape de classification, nous avons donc besoin d'une phrase. Le jeton [CLS] est utilisé pour faire cette étape. Dans la tâche ou nous devons prédire la phrase suivante, il nous faut un moyen de communiquer au modèle ou se termine la première phrase et où commence la deuxième phrase de notre modèle. Le jeton [SEP] va donc servir de délimiteur.

Le modèle BERT va travailler sur une longueur de phrase fixée depuis l'entrée. Selon le texte, certaines phrases seront plus courtes que d'autres donc nous devrons compléter leur longueur grâce à un jeton de rembourrage [PAD].

Le modèle BERT a été pré-entraîné avec des jetons qui avaient tous un identifiant unique. Pour utiliser le modèle pré-entraîné. Il nous faut convertir chaque jeton de la phrase d'entrée en identifiant unique correspondant. Le modèle pré-entrainé étant formé sur un certain dictionnaire, il nous est donc nécessaire pour les jeux d'entrée dont les vocabulaires n'apparaissent pas dans le dictionnaire pré-entraîné de les affectés un jeton spécial [UNK]. Pour ce faire, grâce à l'algorithme WordPiece nous pouvons diviser nos séquences de phrase en plusieurs sous-mots, de sorte que les mots courants répétés puissent être représentés par le modèle.

Par exemple, si nous fournissons au modèle la phrase d'entrée « Nous apprenons le NLP » à la sortie le modèle peut tokeniser notre phrase en la divisant en mot telle que : « Nous », « apprend », « le », « NLP ». A chaque mot sera convertis automatiquement en ID du jeton. La fonction Tokenisation de BERT va donc permettre de diviser une phrase en plusieurs mots en les affectants à des jetons d'identification.

Un exemple du traitement sera sous cette forme :

Pour cette phrase : « Nous apprenons le NLP ».

Tokenisation de la phrase : [‘Nous’, ‘apprenons’, ‘le’, ‘NLP]

Ajouter de [CLS] et [SEP] : [‘[CLS]’, ‘Nous’, ‘apprenons’, ‘le’, ‘NLP’, ‘[SEP]’]

Étape de rembourrage : [‘[CLS]’, ‘Nous’, ‘apprenons’, ‘le’, ‘NLP’, ‘[SEP]’, ‘[PAD]’]

Étape de conversion en ID : [99, 101, 106, 3879, 2, 664, 509].

- ✓ **RoBERTa[2] :**

Le modèle RoBERTa est une amélioration de l’approche BERT qui a été réalisé par Yinhan Loi, Myle Ott, Naman Goyal, Jingfei DU, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov. Il se base sur le modèle BERT publié par Google en 2018 et modifie les hyper-paramètres en le rendant plus robuste.

Le choix des hyperparamètres a un impact sur les modèles finals.

- ✓ **La fonction JACCARD[3] :**

La fonction JACCARD est basée sur une mesure de la similarité en python. En langage mathématique elle se rapproche de la mesure de la distance claudienne. L’intersection de Jaccard correspond à la taille de l’intersection divisée par la taille de l’union de deux ensembles.

<u>Exemple tiré de la source cité en référence :</u>

Sentence 1: AI is our friend and it has been friendly

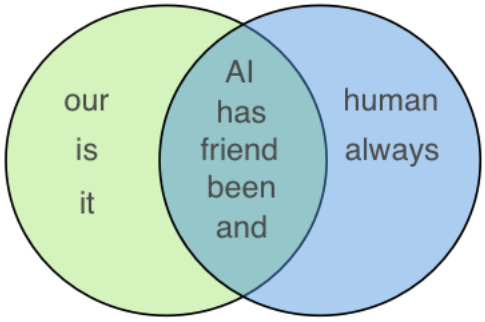
Sentence 2: AI and humans have always been friendly.

[2] Huggingface, « RoBERTa », consulté le 17 novembre 2021, disponible sur : https://huggingface.co/transformers/model_doc/roberta.html.

[3] Sanket Gupta, “Overview of Text Similarity Metrics in Python”, consulté le 17 novembre 2021, disponible sur : https://towardsdatascience.com/overview-of-text-similarity-metrics-3397c4601f50.

Pour comprendre la similarité entre ces deux phrases à l'aide de la similarité de JACCARD. Comme sur l'image.

Figure 6 : Diagramme de Veen des deux phrases pour la similarité de jaccard



Source: Overview of Text Similarity Metrics in Python.

Pour ces deux phrases, la similarité de Jaccard est de $\frac{5}{(5+3+2)} = 0{,}5$. Ce calcul correspond à la taille de l'intersection de deux mots divisés par la taille totale de l'ensemble qui le compose.

III. RÉSULTATS & CONCLUSION :

Pour atteindre les objectifs du projet, nous avons testé plusieurs modèles. Nous avons appliqué le modèle de similarité entre les tests qui nous a permis d'obtenir un résultat 0,0601(annexe, modèle 1). Nous pensons que ce modèle n'était pas très précis. Ce qui nous a conduit à estimer d'autres modèles dans l'optique d'avoir une meilleure précision.

Pour cela, nous nous sommes donc basés sur le fichier BERT & RoBERTa qui sont des dictionnaires pré-entraînés sur wikipédia et BooksCorpus. À l'aide de ces fichiers, nous avons pu appliquer les principes du modèle BERT qui consiste à fournir au modèle un point d'entrée qui sont les fichiers pré-entraînés au format "json".

Nous avons fait à ce niveau trois modèles, un avec BERT et deux modèles avec xlm-RoBERTa où après soumission le meilleur modèle entre les trois est le modèle obtenu grâce au fichier dictionnaire xlm-RoBERTa-large-squad2 avec un score 0,57(annexe, modèle 4). Nous sommes conscients de la possibilité d'amélioration de l'apprentissage de nos modèles par réglage de nos

hyper paramètres. A titre d'exemple, le meilleur score Jaccard obtenu pour ce projet est de 0,831.

Nos plus grandes difficultés dans ce projet ont été tant dans l'apprentissage d'une nouvelle langue or nous sommes toujours novices qu'autant dans la compréhension du sujet.

Un des problèmes qui nous fait perdre du temps, est lié à "cuda" dont nous ne disposons pas sur nos ordinateurs portables. Nous n'avons pas pu réaliser le projet depuis "jupyter notebook". In fine, nous nous sommes contraints à nous tourner vers la plateforme "kaggle" en utilisant son interface "notebook" et surtout paramétrer sur l'accélérateur GPU pour pouvoir prendre en compte "cuda".

IV. ANNEXE :

- ✓ Modèle 1 : Prévision du modèle de Jaccard

```
Average Jaccard Score: 0.06096577470006555
CPU times: user 492 ms, sys: 2.68 ms, total: 495 ms
Wall time: 497 ms
```

	id	PredictionString
0	22bff3dec	7 सितंबर
1	282758170	51 सहित
2	d60987e0e	१८२४ -
3	f99c770dc	13 ஆகும்.
4	40dec1964	2002 இயற்றியது.

- ✓ Modèle2 : Prévision du modèle réalisé avec le dictionnaire BERT

	id	PredictionString
0	22bff3dec	येलन चीन
1	282758170	11 फ़रवरी 2010
2	d60987e0e	१२ मार्च १८२४
3	f99c770dc	13
4	40dec1964	ரோக்டேல் பயனீர்

✓ Modèle 3 : Prévision du modèle réalisé avec le dictionnaire xlm-roberta-base-squad2

	id	PredictionString
0	22bff3dec	येलन चीन
1	282758170	\n4 नवम्बर 2009
2	d60987e0e	१७ अक्टूबर १८८७)
3	f99c770dc	13
4	40dec1964	சுவாமிநாதன் மற்றும் வர்கீஸ் குரியன்

✓ Modèle 4 : Prévision du modèle réalisé avec le dictionnaire xlm-roberta-large-squad2

	id	PredictionString
0	22bff3dec	येलन चीन से हैं। उनकी मां येलन गुट्टा
1	282758170	28 नवम्बर 2007
2	d60987e0e	(१२ मार्च १८२४
3	f99c770dc	13,
4	40dec1964	சுவாமிநாதன்