**Développez une preuve de concept**

PROJET 8 DATA SCIENTIST

**Résumé** : Dans le cadre du projet 6 je devrais trouver les thèmes des questions afin de les proposer aux utilisateurs novices. J’ai utilisé des modèles assez standard. Pour le projet 8, j’ai décidé de traiter cette problématique avec des modèles récents. Je me suis orienté vers les réseaux de neurones et plus particulièrement les Transformers.

Ce document va détailler mon approche et les résultats obtenus.

Voici les différents éléments qui seront détaillés :

* Reprise du projet précédent
* Nouvelle méthode qui sert de Baseline
* Réseau de neurones avec une couche d’embedding
* Méthode moderne pour traiter ce problème : Les Transformers
* Une image contenant LEGO, jouet, encombré, plusieurs

  Description générée automatiquement

WALROFF Jérôme | [jerome.walroff@gmail.com](mailto:jerome.walroff@gmail.com) | 28 mars 2022

Table des matières

[1. Requêtage 3](#_Toc101197770)

[2. Préprocessing des POSTS 3](#_Toc101197771)

[Nettoyage 3](#_Toc101197772)

[3. Modélisation 3](#_Toc101197773)

[3.1. BaseLINE 4](#_Toc101197774)

[Couche de notre modèle 4](#_Toc101197775)

[Lancement du modèle 5](#_Toc101197776)

[Evaluation du modèle 5](#_Toc101197777)

[Exemple sur cinq enregistrements 6](#_Toc101197778)

[3.2. Embedding Modèle 6](#_Toc101197779)

[Couche de notre modèle 6](#_Toc101197780)

[Lancement du modèle 6](#_Toc101197781)

[Evaluation du modèle 7](#_Toc101197782)

[Exemple sur cinq enregistrements 8](#_Toc101197783)

[3.3. Transformers avec T5 9](#_Toc101197784)

[Présentation des Transformers 9](#_Toc101197785)

[Modèle T5 11](#_Toc101197786)

[Lancement du modèle 12](#_Toc101197787)

[Evaluation du modèle 13](#_Toc101197788)

[Exemple sur cinq enregistrements 13](#_Toc101197789)

[3.4. En synthèse 14](#_Toc101197790)

[4) Conclusion 15](#_Toc101197791)

GLOSSAIRE

|  |  |  |
| --- | --- | --- |
| Terme | Description | Lien |
| BERT | Modèle pour traiter le langage. T5 est basé sur BERT | https://www.webrankinfo.com/dossiers/google-search/bert |
| [LSTM](https://www.wikiwand.com/fr/R%C3%A9seau_de_neurones_r%C3%A9currents#/Long_short-term_memory) (Long-Short Term Memory) | Couche d’un réseau de neurones afin de gérer la mémoire d’un neurone. | https://penseeartificielle.fr/comprendre-lstm-gru-fonctionnement-schema/ |
| Transformers | Modèle en réseau de neurones très efficace dans le traitement du langage naturel | https://ledatascientist.com/a-la-decouverte-du-transformer/ |
| T5s | Librairie mettant en œuvre les Transformers | https://huggingface.co/docs/transformers/model\_doc/t5 |
|  |  |  |
|  |  |  |
|  |  |  |

# Requêtage

Dans le projet 6 le jeu de données était proposé par le site Stackoverflow. Via un outil de requêtage de type SQL Nous avions obtenu un jeu de données d’environ 16 000 questions. J’ai modifié un paramètre de sélection afin d’augmenter le jeu de données jusqu’à 50 000 questions.

# Préprocessing des POSTS

Je n’ai pas réutilisé le prepocessing que j’avais mis en œuvre dans le cadre du Projet 6 afin d’utiliser ceux proposés par les différents modèles que je souhaite utiliser.

### Nettoyage

Le seul traitement réutilisé sera celui traitant le champs Tags afin d’en extraire les étiquettes.

Spécificité champs TAGS

* Expression régulière : Celui-ci est encapsulé dans une balise <…>. J’ai donc réalisé une expression régulière pour récupérer l’information contenue à l’intérieur et utiliser la fonction word\_tokenize de NLTK
* L’idée au niveau des Tags était de ne retenir que les 100 plus utilisés. J’ai donc réalisé un comptage de la présence du tag pour constituer un Top100. J’ai alors supprimé tous les tags n’appartenant pas à ce Top100. Ceci a engendré des questions sans Tags que j’ai supprimées.

# Modélisation

J’ai choisi de m’orienter vers des réseaux de neurones pour traiter mon sujet.

Je rappelle que dans le cadre du Projet 6, la catégorisation de question se faisait avec les méthodes suivantes :

* TD-IDF puis NMF (Non negative matrix factorization) pour une approche non supervisée
* Arbre de décision après TF-IDF pour une approche supervisée

Dans le cadre du Projet 8 je vais m’orienter vers les solutions suivantes :

* Un réseau de neurone très simple afin de servir de Baseline
* Un réseau de neurones un peu plus complexe avec une couche d’embedding et LSTM
* L’utilisation de Transformers (T5) qui est une méthode récente

Suivant le modèle utilisé la préparation des étiquettes a été différente. Dans la première partie j’ai utilisé **StringLookup. Cette librairie de Keras permet de créer le vocabulaire de sortie.**

Pour le modèle avec les Transformers je suis sur du One Hot Encoder simple.

## BaseLINE

Avant de lancer mon modèle il faut vectoriser mon information afin de l’exploiter par la suite. J’ai utiliser **TextVectorization** de Keras avec les paramètre suivants :

* Max\_Tokens : Taille de mes mots uniques en entrée
* Ngramms : Nombre de mots associés à partir des mots uniques pour constituer notre modèle
* Output : Ici nous utiliserons TD\_IDF
* Standardize : Traitement réalisé sur nos mots. Ici j’ai choisi de mettre en minuscule et de retirer les ponctuations

Comme vous pouvez le voir j’utilise TD-IDF qui va produire un vecteur assez important. Cette méthode ne sera pas utlisée dans l’Embedding modèle.

### Couche de notre modèle

Pour constituer cette Baseline j’ai constitué un réseau de neurones très simple avec les couches suivantes :

Une image contenant table

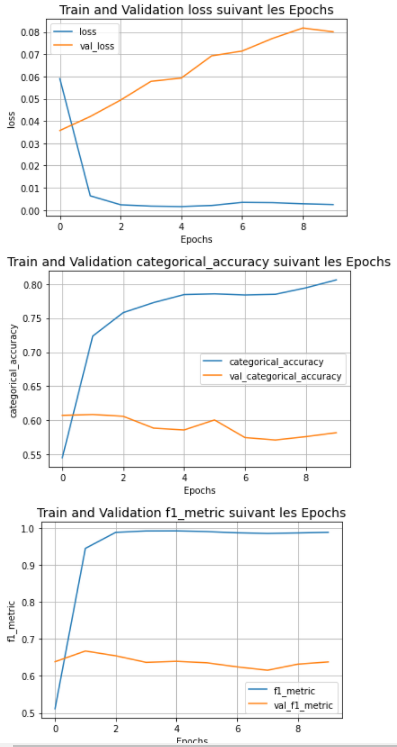
Description générée automatiquement

Nous avons donc deux couches connectées avec ReLu.

La dernière couche est Dense également permettant de sortir nos étiquettes., elle a comme taille de sortie notre nombre d’étiquettes.

### Lancement du modèle

Pour l’apprentissage de notre modèle nous allons spécifier un nombre d’Epoch qui représente le nombre de fois ou le modèle va être joué successivement afin d’en améliorer son résultat. Pour déterminer ce nombre j’ai constitué des courbes pour évaluer certaines métriques et les présenter.  
A travers ces courbes il s’agit de déterminer le nombre d’Epoch entre le gain obtenu et le temps de traitement. Il s’agit toujours d’un compromis entre ces éléments.



**J’ai décidé de lancer l’apprentissage de mon modèle avec 10 Epochs.**

### Evaluation du modèle

Je lancé l’évaluation de mon modèle sur mon jeu de test que j’avais préalablement constitué.  
J’obtiens le score suivant

**Accuracy sur le jeu de test : 61.31%.**

### Exemple sur cinq enregistrements

Et voici les résultats obtenus sur les cinq éléments de mon jeu de données de testUne image contenant texte

Description générée automatiquement

Les résultats sont plutôt bons.

## Embedding Modèle

Dans ce modèle la vectorisation avec TF-IDF n’avait pas de sens. J’ai opté donc pour la création de vecteurs avec 300 dimensions.

Je demande toujours la mise en minuscule et la suppression de la ponctuation.

### Couche de notre modèle

J’ai ici les couches suivantes :

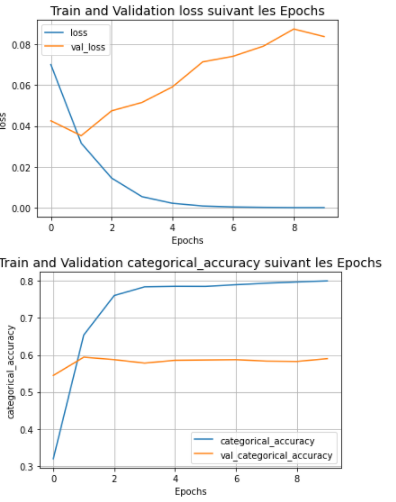
* Une couche d’entrée InputLayer
* Une couche d’Embedding
* Une couche LSTM
* Une couche Flatten pour mettre à plat ma matrice
* En enfin la couche Dense qui va prédire les étiquettes

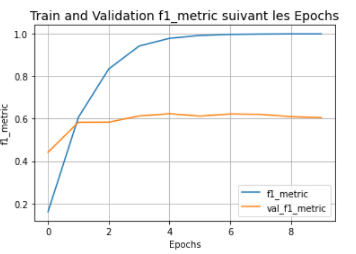
Une image contenant table

Description générée automatiquement

### Lancement du modèle

Voici les résultats que j’ai obtenus





### Evaluation du modèle

Je lancé l’évaluation de mon modèle sur mon jeu de test que j’avais préalablement constitué.  
J’obtiens le score suivant



### Exemple sur cinq enregistrements

Et voici les résultats obtenus sur les cinq éléments de mon jeu de données de test

Une image contenant texte

Description générée automatiquement

**Axe d’amélioration de ce modèle :**

Pour le rendre beaucoup plus efficace j’aurai dû utiliser dans ma couche d’embedding un langage pré-entrainé afin de constituer les liens entre les mots de mon vocabulaire.  
J’aurai pu me tourner vers Glove.  
Par faute de temps je n’ai pas pu le mettre en œuvre.

## Transformers avec T5

### Présentation des Transformers

Le traitement automatique du langage naturel est, aujourd’hui, essentiellement dominé par les modèles séquence à séquence (ou seq2seq). Voila pourquoi dans le cadre de ce projet j’ai voulu m’orienter vers ce genre de solution.

Ce type de modèle est particulièrement adapté à la langue et à plusieurs tâches de NLP classiques telles que la traduction de texte. Il correspondait donc parfaitement à ma problématique.

Parmi les modèles seq2seq, un en particulier s’est imposé. Il s’agit du Transformer.

Un modèle seq2seq prend en entrée une séquence (suite d’éléments du même type) et renvoie une séquence en sortie. Il est très largement utilisé dans le domaine des traductions.

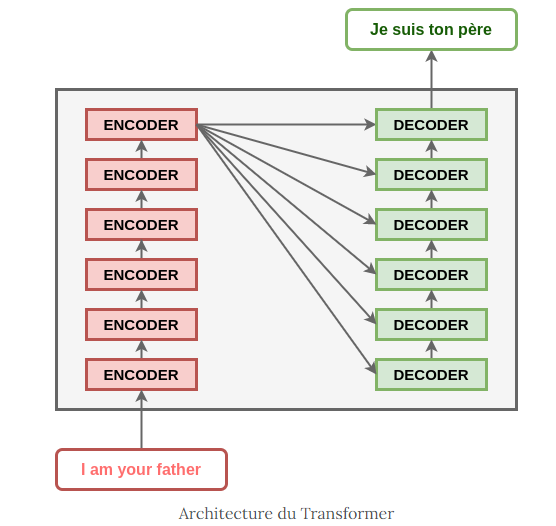
Une image contenant texte

Description générée automatiquement

Pour faire des modèles séquence à séquence avant la venue du Transfomer, il fallait avoir recours au fameux [LSTM](https://www.wikiwand.com/fr/R%C3%A9seau_de_neurones_r%C3%A9currents#/Long_short-term_memory) (Long-Short Term Memory). Il s’agit d’un réseau de neurones plus élaboré qui permet de conserver ou supprimer les informations du texte initial. En d’autres termes cela permet de donner du sens aux mots utilisés. Je ne vais pas détailler davantage le LSTM qui est aussi complexe que ces résultats sont performants.

Le rôle du LSTM est de prendre en compte l’interdépendance des mots. Mais ce modèle a une limitation : il est relativement lent à entraîner et très peu parallélisable. Du coup dès l’instant que nous avions des phrases longues les performances chutaient grandement.

L’architecture du Transfomer a hérité du pattern Encoder-Decoder. La partie « encodage » contient 6 encodeurs montés l’un après l’autre. La partie « décodage » consiste en 6 décodeurs également montés l’un après l’autre mais prenant chacun, comme entrée supplémentaire, la sortie du 6ème encodeur.



**Qu’est-ce que l’Attention :**

Les modèles d’attention essaient de traduire une partie de la phrase à la fois, ce qui le rend plus efficace. L’idée principale et de permettre au décodeur de regarder en arrière et d’extraire les informations importantes qui sont utiliser au décodage.

Le mécanisme d’attention consiste à mesurer si deux éléments de deux séquences sont proches. Au sein d’une même séquence nous allons parler de self attention ou auto attention. Ainsi chaque encodeur et décodeur met en œuvre ce mécanisme d’attention.

Les Transfomers ont été une belle avancée à leur sortie car beaucoup plus rapide à prédire et avec de meilleurs résultats. Les transformers sont sortis en 2017 grâce à Google.

Bert (Bidirectional Encoder Representations from Transformers) est l’un des modèles les plus connus. Bert va masquer aléatoirement des mots dans la séquence puis il essaie de la prédire. Il va fonctionner de gauche à droite et de droite à gauche prenant en compte les mots avant et après. Bert était une méthode pour pré entrainer les encodeurs. Il fallait avoir recours à des décodeurs afin d’en obtenir les résultats. De plus Bert travaillait uniquement avec des modèles non supervisés.

**Pourquoi T5**

J’ai choisi de m’orienter vers T5 (The Text to text transfer transformer) avec la mise en place d’un Tutoriel que j’ai trouvé dans un contexte de multi label. Dans de nombreux benchmarks T5 obtient des résultats supérieurs à Bert et est plus récent. De plus il permet de travailler avec des modèles supervisés.

Voici le tutoriel utilisé dans le cadre de mon projet

<https://www.kaggle.com/code/prithvijaunjale/t5-multi-label-classification/notebook>

Tous les problèmes seront traités comme des prédictions de texte. T5 est Entrainé sur une grosse quantité de données. C’est un Framework unifié pour traiter tous les problèmes de « text to text », supervisé ou non supervisé.

### Modèle T5

J’ai utilisé la librairie **PyTorch** et **Transformers T5**

Afin de préparer mes étiquettes pour le modèle j’ai utilisé MultiLabelBinarizer

Et comme dans l’exemple précédent je prépare

* Un jeu d’entrainement
* Un jeu de validation
* Un jeu de test

**Pre traitement**

Ici je dois mettre en place ma propre fonction de traitement du texte.

Les traitements suivants sont appliqués à mon texte entrant :

* Suppression des tabulations et des retours chariots
* Suppression des ponctuations

Au niveau de mes étiquettes je dois mettre en place un traitement qui gère une chaine de caractère. En effet la méthode présentée va prédire une chaine de caractère représentant les étiquettes. Je vais donc constituer ces mêmes chaines de caractères afin d’alimenter mon modèle.

Le modèle T5 prends deux éléments en entrée

* src\_input\_ids : Il s’agit du texte entrée qui doit permettre de prédire l’étiquette. Dans notre contexte il s’agit du titre et du body de notre question.
* tgt\_input\_ids Il s’agit des étiquettes de sortie que l’on veut prédire. Ici nous avons 100 étiquettes possibles. Cependant nous pouvons avoir plusieurs étiquettes pour une même prédiction.

Afin de constituer les datasets attendus par le modèle, une class a été créée T5Dataset. Celle-ci va renvoyer tous les éléments nécessaires à notre traitement.

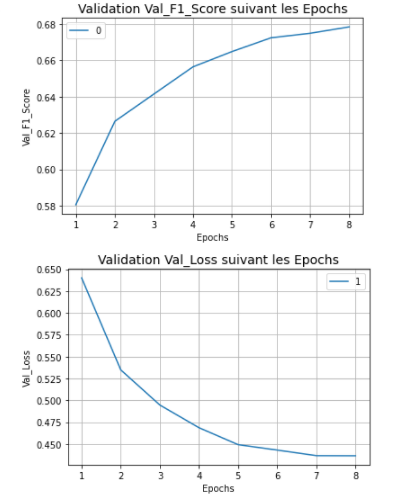
A partir de ces Dataset un autre objet doit être créé à partir de la fonction DataLoader. Celle-ci prendre en entrée un Dataset afin de retourner un torch.utils.data.dataloader.DataLoader

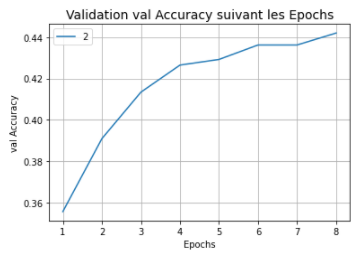
Enfin une autre classe a été implémentée T5Model afin de créer le modèle T5 en tant que tel.

A l’intérieur d’un Epcoh nous découpons notre jeu de test sur des sous périmètre afin d’évaluer notre entrainement et notre validation.

### Lancement du modèle

Voici les courbes présentées pour 8 Epochs





Nous pourrions continuer d’augmenter le nombre d’Epoch mais le traitement dure déjà très longtemps et nous constatons que les gains ne sont plus très importants.

### Evaluation du modèle

Voici les résultats obtenus



### Exemple sur cinq enregistrements

Une image contenant texte

Description générée automatiquement

**Les résultats sont exceptionnels car toutes les prédictions sont justes**.

## En synthèse

Voici le tableau de synthèse. J’ai utilisé la librairie de Sklearn pour avoir des calculs similaires dans mes différents modèles. Nous pouvons donc constater une accuracy plus faible qu’auparavant pour la Baseline et le modèle Embedding. La librairie de Sklearn calcul l’accuracy avec une stricte égalité.

De plus dans le cadre de ma Baseline et modèle Embedding cette accuracy est uniquement sur le Top 1. Ce qui rend encore plus le résultat du Transformer exceptionnel.

|  |  |  |
| --- | --- | --- |
| Modèle | F1 Score | Accuracy |
| Arbre de décision après TFD-IDF (Ancien projet) | 0,562 | 33,29% |
| Réseau de neurones simple | 0,571 | 35 ,84%(TOP 1) |
| Avec Embedding | 0,601 | 43,75%(TOP 1) |
| Transformers T5 | 0,712 | 50% |

Les résultats du Transformers sont bien au-dessus de tous les autres.

De plus avec les Transformers je n’ai pas à me soucier du seuil retenu pour valider ou non l’étiquette (Probabilité de la prédiction de l’étiquette). Ce qui est encore un gain non négligeable.

# Conclusion

Ce projet m’a particulièrement plu car j’ai pu reprendre une ancienne démarche pour évaluer une méthode plus moderne. Cela permet de comprendre rapidement les progrès dans ce domaine et l’enjeu pour un Datascientist de toujours être à l’écoute des solutions.

Les résultats obtenus avec les Transformers ont été les plus performants et sont exceptionnels sur l’échantillon de 5 enregistrements que j’ai sélectionnés.

Permettre de traiter tous les sujets avec la même librairie est une vraie facilité. Néanmoins la modernité de la librairie et sa complexité de mise en œuvre ont été un vrai défi. En effet il n’y avait pas beaucoup de ressources sur le sujet, et j’ai trouvé la documentation en ligne peu développée.

Une fois appréhendée, transposer à une autre problématique de traduction, par exemple, devrait s’avérer beaucoup plus simple.