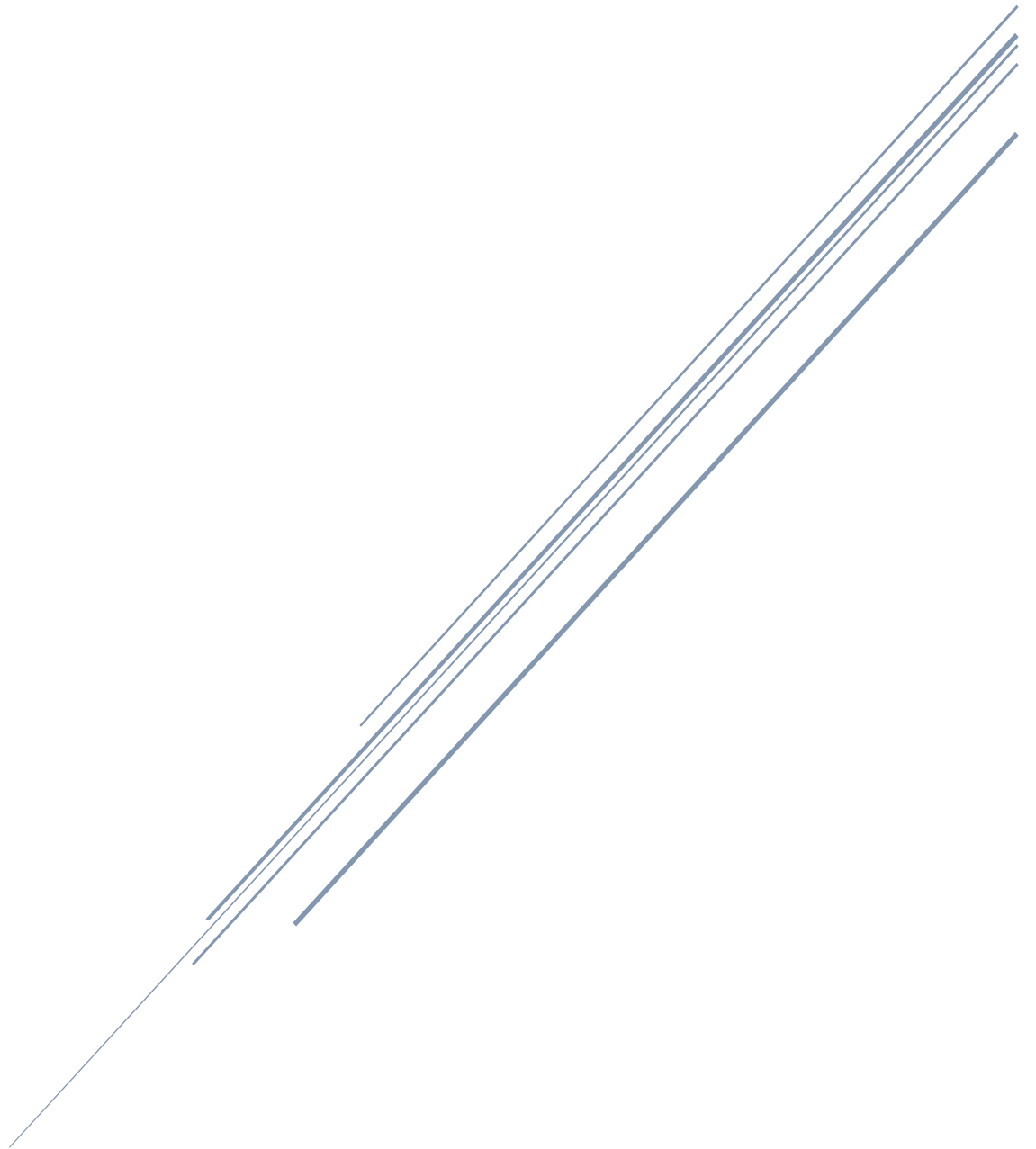


PARTICIPEZ A UNE COMPETITION KAGGLE !

NBME - Score Clinical Patient Notes



Thibaud GROSJEAN
OpenClassrooms – Ingénieur Machine Learning

SOMMAIRE

<i>Introduction</i>	<i>3</i>
<i>Compétition.....</i>	<i>3</i>
<i>Environnement</i>	<i>5</i>
<i>Données</i>	<i>6</i>
<i>Présentation du Modèle.....</i>	<i>7</i>
<i>Implémentation.....</i>	<i>8</i>
<i>Amélioration de la Ressource Existante</i>	<i>9</i>
<i>Amélioration du Score</i>	<i>10</i>
<i>Conclusion</i>	<i>11</i>

INTRODUCTION

Kaggle est une plateforme web de compétitions de *Data Science* créée en 2010 par Anthony Goldbloom et appartenant à *Google* depuis 2017. Les organisations (entreprises ou associations, groupes scolaires) fournissent les données, règles et -optionnellement- les lots des compétitions. Les participants entraînent des modèles de *machine learning* afin d'obtenir le meilleur score. Pour se faire, la plateforme met à la disposition des participants une puissance de calcul (*processeurs graphiques pour l'entraînement des modèles*) par le biais de ses *notebooks* (*IDE de type Jupyter*).

Outre l'aspect compétitif de la plateforme, *Kaggle* offre aussi un aspect collaboratif : elle propose par exemple aux participants de partager leur code (*notebooks ayant permis d'obtenir un score pour une compétition*) publiquement, offre des cours de *Data Science* gratuits ainsi qu'un forum de discussion ouvert et assume son slogan : « The Home of Data Science » (« La Maison des Sciences de la Donnée »).

Nous avons participé à une compétition active afin d'obtenir un score et de partager des ressources avec la communauté.

COMPETITION

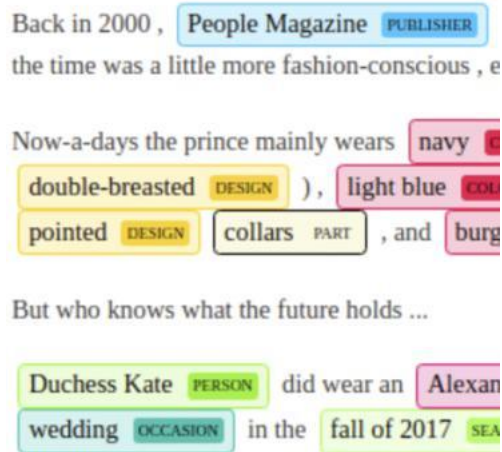
La compétition [*NBME - Score Clinical Patient Notes*](#) a été créée par le *National Board of Medical Examiners*, association à but non lucratif fondée en 2015 qui développe des examens pour les professionnels de la santé aux Etats-Unis.

Les données sont issues du *USMLE® Step 2 Clinical Skills examination*, un examen de compétences cliniques qui permettait aux étudiants et diplômés de médecine de pouvoir pratiquer légalement aux Etats-Unis. L'examen a été supprimé définitivement du parcours d'obtention de licence de médecine en février 2021 en raison de la crise sanitaire.

Lors de cet examen -une simulation de consultation- les candidats étaient mis face à des professionnels de santé jouant le rôle de patients standards. Les candidats étaient évalués sur leur capacité à constituer un historique médical exhaustif du patient factice.

Pour cette compétition, le *NBME* fournit des notes (historiques médicaux relevés par les candidats). L'objectif de cette compétition est de fournir un modèle de *machine learning* capable de détecter dans les notes des candidats les portions de texte faisant référence aux différents symptômes du patient standard et de les nommer.

Il s'agit donc d'une problématique de *traitement du langage naturel* (*Natural Language Processing*), et plus spécifiquement de *reconnaissance d'entités nommées* (*Named Entity Recognition*).



Exemple de reconnaissance d'entités nommées

Pour chaque *entité nommée* cible de chaque *document* cible, le modèle parfait sera capable de prédire à quel endroit dans le *document* l'*entité nommée* en question apparaît (*indices ou slice des caractères de la string*). Dans l'exemple ci-dessus, l'entité « *People Magazine* » débute à l'indice (*index*) 15 et se termine à l'indice 28.

Le score est calculé grâce à la formule *F1 micro-moyenné (micro-averaged F1 score)*. Pour chaque *entité nommée* de chaque *document (sample)*, on calcule le nombre :

- d'indices *Vrais Positifs (TP)* : les indices *vrais (ground truth)* prédits correctement par le modèle,
- de *Faux Négatifs (FN)* : les indices *vrais* que le modèle a ratés,
- de *Faux Positifs (FP)* : les indices prédits par le modèle et qui n'auraient pas dû l'être.

On moyenne ces 3 nombres pour obtenir un *score F1 (voir la formule ci-dessous) micro-moyenné*.

$$\frac{tp}{tp + \frac{1}{2}(fp + fn)}$$

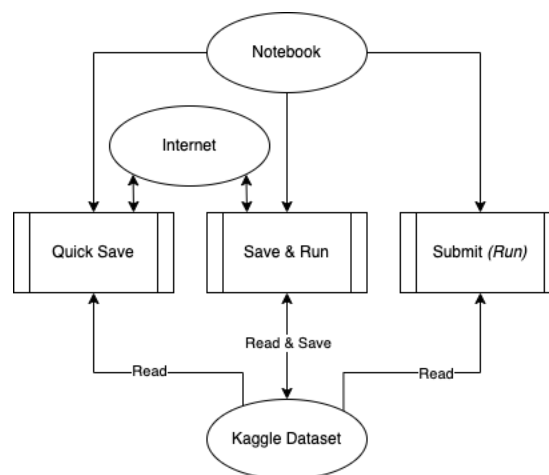
Formule du score F1

ENVIRONNEMENT

Afin de participer à la compétition, il est obligatoire d'utiliser *Kaggle Notebooks* pour effectuer les soumissions. Le *notebook* doit générer un fichier nommé *predictions.csv* afin que les résultats soient évalués. Dans les faits, lors de la soumission, le *notebook* est exécuté sur les serveurs de *Kaggle* afin de générer le fichier de prédictions et de calculer le score.

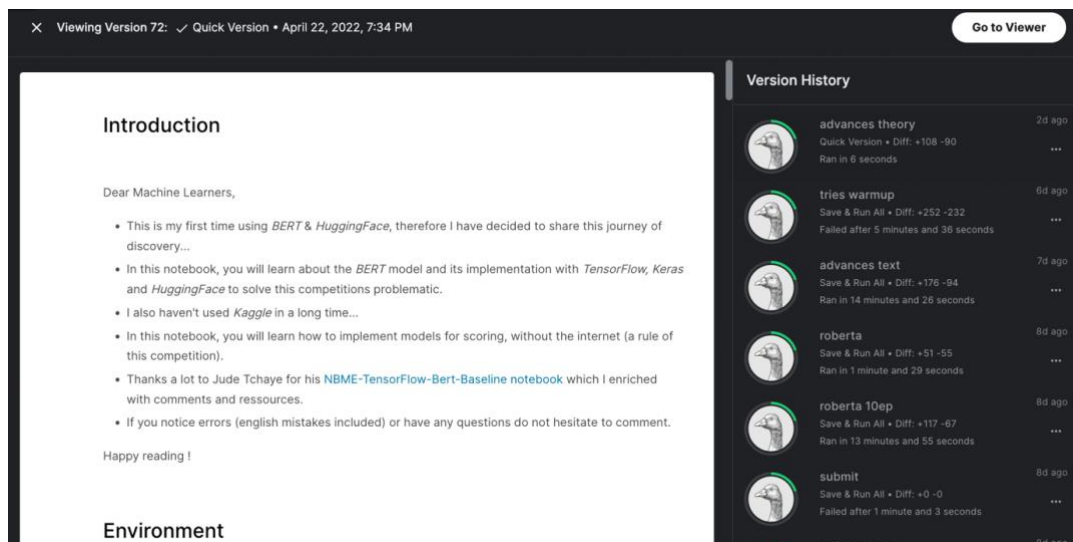
Autre spécificité importante : lors de la soumission, le *notebook* n'a pas accès à internet (*l'option doit être désactivée*). Par conséquent, le *notebook* ne peut pas se connecter à internet pour, par exemple, télécharger des modèles pré-entraînés. Pour utiliser des données sans les télécharger directement depuis le *notebook*, il est tout de même possible de les faire persister sur les serveurs de *Kaggle* sous la forme d'un *dataset* afin de les charger pendant l'exécution.

Pour se faire, nous avons créé un *dataset Kaggle* alimenté par les *Outputs* de notre *notebook*. Lorsque la méthode « *Save & Run All* » est utilisée pour sauvegarder une nouvelle version du *notebook*, le *dataset* est mis à jour avec les fichiers générés pendant l'exécution. En ajoutant ce *dataset* aux *Inputs* du *notebook*, l'instance d'exécution peut lire les fichiers directement sur les serveurs *Kaggle* ce qui permet de se passer d'internet.



Environnement Kaggle Notebooks

Kaggle Notebooks intègre un logiciel de versionnage des fichiers similaire à *Git*, avec une interface simple et intégrée. Deux options de sauvegarde (*commit*) sont offertes à l'utilisateur : il peut choisir une sauvegarde sans exécution, ou une sauvegarde avec exécution (dans ce cas-là, une exécution est lancée dans une instance du serveur, et les fichiers persistants du *dataset* sont mis à jour). Nous utiliserons cette solution pour enregistrer les poids du modèle, le *Tokenizer* et les données générées par notre *kernel*.

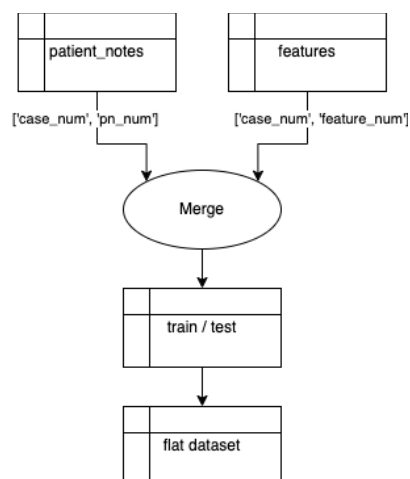


Aperçu du système de versionnage de Kaggle Notebooks

Dernière spécificité importante : lorsque l'option *GPU (processeur graphique)* est activée, *Kaggle* fournit plusieurs *GPUs* (les *GPUs* supplémentaires sont appelés des répliques) dont l'utilisation doit être parallélisée. Cela implique, entre autres, d'appliquer une stratégie de parallélisation et d'utiliser des générateurs pour alimenter en données le modèle pendant l'entraînement.

DONNEES

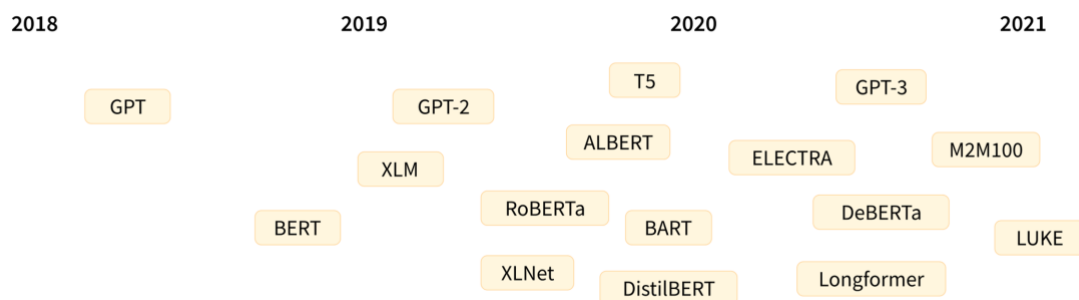
Les données sont fournies dans une forme *relationnelle* : 3 fichiers *.csv* distincts dont *train* qui comporte l'ensemble des *clés primaires*, *features* qui liste les 143 classes de symptômes et *patient_note* qui référence les *documents*. Pour constituer un *fichier plat (dataset)* nous avons appliqué des *jointures* (voir la figure ci-dessous). Nous avons obtenu un fichier plat composé de 14300 *samples*.



Pipeline de preprocessing

PRESENTATION DU MODELE

Le modèle *BERT* (*Bidirectional Encoder Representations from Transformers*) de Google décrit dans l'article *Attention Is All You Need* est l'un des premiers modèle *Transformeur*, après la sortie de *GPT* en 2017.



Chronologie du développement des Transformeurs

BERT est un modèle *auto-attentif*, ce qui lui permet de développer une compréhension statistique du langage avec lequel il a été entraîné sans requérir d'étiquetage des données (*Data Labelling*). Autrement dit, ce type de modèle peut être entraîné de manière *auto-supervisée* (*self-supervised learning*). Cela représente un avantage considérable dans le domaine du *NLP* où la création de ce type de *jeu de données* représente un travail considérable pour l'homme.

Le modèle a été pré-entraîné (*pre-training*) avec de larges quantités de données (*Wikipédia*) sur des tâches de *Causal Language Modeling* et de *Masked Language Modeling*, et ce simultanément, afin de développer une véritable compréhension de la langue anglaise. *BERT* a ainsi atteint des performances de pointe (*state-of-the-art*) dans de nombreuses tâches de *NLP*, ce qui représente une avancée considérable dans ce domaine.

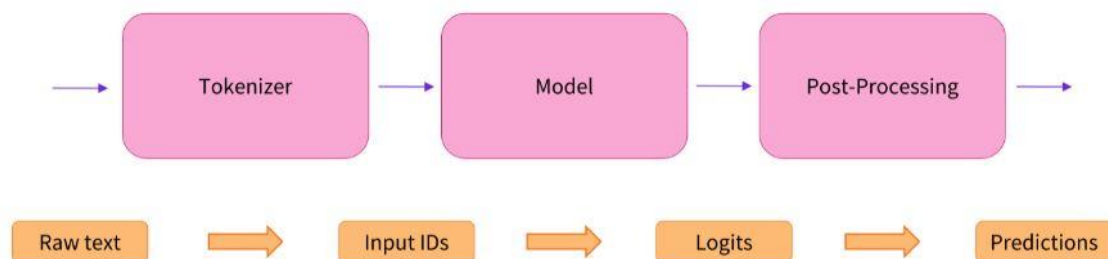
Nous avons spécialisé (*fine-tuning*) une variante de *BERT* : le modèle *RoBERTa* (*base*) de Facebook afin de générer des prédictions pour la compétition. Selon Facebook, « *RoBERTa* itère sur la procédure de *pre-training* de *BERT*, ce qui inclut un entraînement plus long, des *batches* plus de plus grande taille ; le retrait de l'objectif de *Next Sentence Prediction* (prédiction de la phrase suivante) ; l'entraînement avec des séquences plus longues ; ainsi qu'un *pattern* de changement dynamique des masques appliqué sur les données d'entraînement ».

IMPLEMENTATION

Nous avons implémenté le modèle à l'aide du langage *Python*, des API *TensorFlow (Keras)* et *transformers (HuggingFace)*.

HuggingFace fournit, entre autres, l'API *transformers* qui facilite l'implémentation de *Transformeurs* pré-entraînés, ainsi qu'un *repository (huggingface.co)* des différents modèles ; ces outils permettent le téléchargement de modèles et leur import dans un *kernel Python*, sous *TensorFlow* ou *Pytorch*. L'API permet aussi d'importer le *Tokenizer* et sa configuration pour appliquer l'ensemble des étapes de *preprocessing* directement sur des données non traitées (texte brut) : le *Tokenizer* applique une tokenization *end-to-end* à l'aide de modèles tels que la *Wordpiece Tokenization* (dans le cas de *RoBERTa*, il s'agit du modèle *BPE*). Le *Tokenizer* permet également d'inverser les *logits* du modèle afin d'obtenir des prédictions lisibles par l'homme.

En somme, le *pipeline* des *Transformeurs* est simple en apparence et à l'utilisation, grâce au *Tokenizer* qui encapsule l'ensemble des transformations nécessaires pour entraîner le modèle (voir la figure ci-dessous).



Pipeline des Transformeurs

Afin de spécialiser le modèle pour la tâche de *Named Entity Recognition*, nous avons dû fournir au modèle les *embeddings* appropriés. Les *embeddings* sont des vecteurs numériques représentant les caractéristiques des données textuelles de chaque *document*.

Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	##ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	E_{ing}	$E_{[SEP]}$
	+	+	+	+	+	+	+	+	+	+	+
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
	+	+	+	+	+	+	+	+	+	+	+
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

Exemple d'embeddings pour les Transformeurs

Dans notre implémentation, nous avons utilisé les *embeddings* suivants :

- *Input ids (Tokens)* : encodés avec le *Tokenizer* associé,
- *Attention mask* : qui pointent vers le texte dont nous essayons de reconnaître les entités,
- *Cible* : les portions du texte correspondant aux symptômes ont été encodés grâce à un *LabelEncoder*, ces *embeddings* sont fournis au modèle pour *y*,
- nous n'avons pas utilisé les *embeddings de segment* (qui dénote par exemple le début ou la fin d'une phrase).

Nous avons utilisé l'*API fonctionnelle* de *Keras* afin de construire notre modèle de machine learning. Dans le cadre de cette application de *transfer learning*, nous avons construit les *input layers* ainsi qu'un *output layer* pour intégrer le modèle *pré-entraîné* et générer un modèle utilisable.

AMELIORATION DE LA RESSOURCE EXISTANTE

Afin de répondre aux exigences du projet sur la réutilisation de ressources existantes, nous avons pris pour base le *notebook NBME-TensorFlow-Bert-Baseline* de Jude TCHAYE. Ce *notebook* est une implémentation du modèle *BERT* avec *TensorFlow (Keras)* et *transformers (HuggingFace)* pour répondre à la problématique de la compétition.

Ce *notebook* contient très peu de texte et de commentaires, c'est donc la première amélioration que nous avons réalisée : nous avons créé une ressource didactisée. Pour cela, nous avons rédigé une partie théorique incluant de nombreuses références à destination de ceux qui débutent avec ce type de modèle et ajouté des commentaires dans les cellules de code afin d'en faciliter la compréhension.

De plus, nous avons :

- Décrit les données brutes et leurs transformations afin que l'utilisateur comprenne comment le *fichier plat* a été généré avant la *tokenisation* des documents.
- Simplifié l'utilisation du *notebook* en mettant en avant les différents paramètres (variables modifiables par l'utilisateur, tel que le lien vers le modèle *HuggingFace* et les *principaux hyperparamètres*) afin que l'utilisateur puisse s'appropriier le *notebook* et effectuer des soumissions aisément après son clonage. Entre autres, nous avons développé une fonction qui fournit des informations sur l'état des données et sur les étapes à compléter pour pouvoir effectuer la soumission.
- Amélioré la mise en forme du code dans le style de programmation *PEP8*.

AMELIORATION DU SCORE



NBME-TensorFlow-Bert-Baseline

Updated 3mo ago

Score: 0.721 · 6 comments · NBME - Score Clinical Patient Notes +1

44

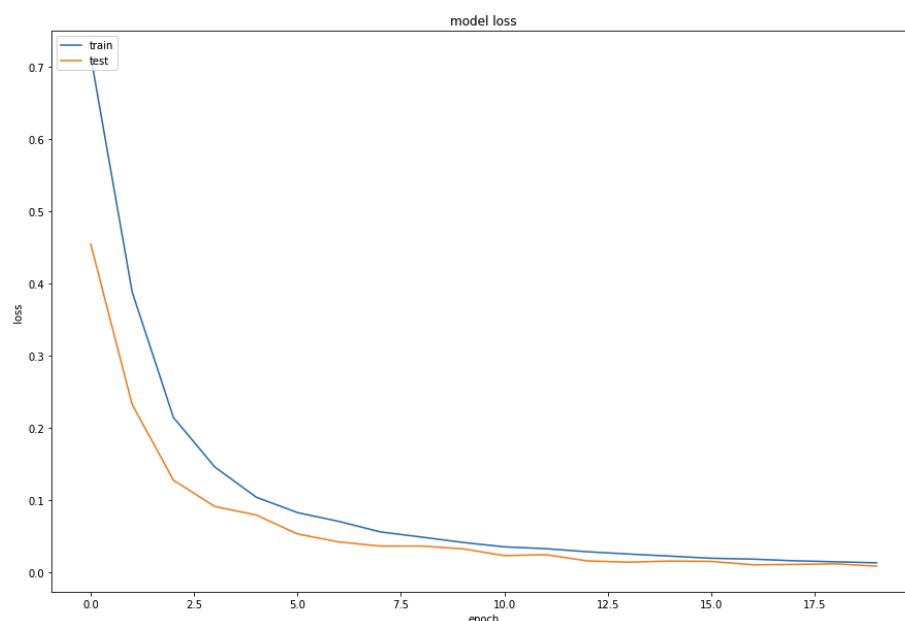
Silver ...

Score initial (notebook originel)



Pour obtenir le score du *notebook* originel (score F1 micro-moyenné de 0.721), nous avons :



- Réglé de nombreux messages d'alerte en rapport avec la parallélisation afin d'être certain de sa bonne mise en œuvre, en implémentant notamment des générateurs pour fournir les données au modèle lors de son entraînement avec des répliques.
- Effectué un *split* des données, afin d'optimiser le modèle et le nombre *d'epochs* grâce un *dataset* de validation.
- *Pour ces deux derniers points, il a notamment fallu convertir les datasets d'entraînement et de validation au format TensorFlow, utiliser les méthodes Tensorflow natives pour effectuer le split des données, ainsi que de calculer les steps per epoch.*
- Créé des visualisations de l'entraînement du modèle afin de de s'assurer qu'il merge correctement. Implémenté un *model checkpoint* pour obtenir le modèle le plus performant.
- Implémenté un modèle plus performant (RoBERTa) que le modèle initialement proposé (BERT).

En ajustant le *learning rate* nous avons obtenu un modèle qui merge parfaitement (voir figure ci-dessous).



Model loss par epoch

998	Thibaud GROSJEAN		0.832	23	3d
	Your Best Entry! Your most recent submission scored 0.832, which is an improvement of your previous score of 0.747. Great job!				Tweet this

1092	Thibaud GROSJEAN		0.843	24	44m
	Your Best Entry! Your most recent submission scored 0.843, which is an improvement of your previous score of 0.832. Great job!				Tweet this

Score amélioré

CONCLUSION

Au fil de ce projet, nous avons pu prendre en main les *Kaggle notebooks* afin de participer à la compétition, en utilisant des ressources de la communauté, découvrir les *Transformeurs* et les modèles de type *BERT* afin d'améliorer notre score, expérimenter avec la parallélisation de *GPUs*, et finalement... Partager nos découvertes, ainsi que des articles généralistes et techniques avec la communauté.

Bien sûr, de nombreuses améliorations restent possibles, nous pourrions par exemple implémenter une recherche d'hyperparamètres plus avancée, une *cross-validation*, utiliser des modèles plus lourds et plus performants tel que le *RoBERTa large*.

- Lien vers la compétition : [NBME - Score Clinical Patient Notes](#)
- Lien vers le *notebook* originel : [NBME-TensorFlow-Bert-Baseline](#)
- Lien vers notre version : [TensorFlow-Transformers-Baseline-For-Beginners](#)