Essais apprentissage automatique

Predictions

- Je vais essayer de prédire les pauses entre les bursts, donc des valeurs continues.
- Les modèles qui peuvent être utilisés pour ce type de valeurs sont des modèles de régression comme régression linéaire, arbre de décision et random forest etc…
- Le premier challenge était d’adapter les données pour qu’elles soient utilisables par ce type de modèle.

Data

- La régression linéaire et la plupart des modèles traditionnels de ML demandent que les inputs soient dans un format tabulaire plat.
- Cependant, dans notre cas nous avons plusieurs lignes par burst :

4	971.839	21.78	8.49	30.27	0.72	0.28	100	97	94	3	3	76	128	171	P	Beaucoup␣critiqué␣cette␣dernière␣s'avère␣parfois␣ene
4	971.839	21.78	8.49	30.27	0.72	0.28	100	97	94	3	3	128	127	171	P	⌫
4	971.839	21.78	8.49	30.27	0.72	0.28	100	97	94	3	3	127	170	171	P	␣effet␣fort␣efficace␣sur␣certaines␣points,␣
4	971.839	21.78	8.49	30.27	0.72	0.28	100	97	94	3	3	170	169	171	P	⌫
4	971.839	21.78	8.49	30.27	0.72	0.28	100	97	94	3	3	169	168	171	P	⌫
4	971.839	21.78	8.49	30.27	0.72	0.28	100	97	94	3	3	168	170	171	P	.␣

Data

- De plus, la colonne charBurst contient des données textuelles que la régression linéaire ne peut pas gérer directement. Il faut donc les pré-processer pour obtenir des valeurs numériques qui représentent le texte. Il y a plusieurs techniques comme one-hot encoding, TF-IDF ou bien les embeddings.

- Cela s'applique aussi aux informations sur les pos et les chunks. Nous avons utilisé Word2Vec. On obtient pour chaque mot une liste de 100 vecteurs permettant de représenter au mieux ce dernier. Nous remplaçons ensuite les colonnes concernées par leurs équivalents vectorisés.
- L'autre problème est la longueur variable de chaque burst. Les bursts peuvent avoir différents nombres de caractères. Gérer des données à longueur variable dans les modèles traditionnels peut être complexes sans plusieurs étapes de pre-processing.

Solutions possibles :

1 - Feature Engineering : plutôt que d'utiliser le texte brut, nous pouvons extraire des features numériques pertinentes de la colonne **charBurst** afin de les utiliser dans notre modèle. Par exemple :

- **Nombre de caractères dans le burst** : la longueur d'une chaîne de caractères dans un burst.
- **Intervalle de position moyen** : moyenne de la différence entre la position de départ et celle de fin.
- **Nombre total de modifications** : compte du nombre de modifications (ex : effacements) dans le charBurst.

2- Aplatir la structure : Convertir la structure imbriquée en une structure plate où chaque burst est représenté par ses caractéristiques agrégées. Cela sera plus facile à utiliser avec des modèles comme la régression linéaire.

3 - Modèles avancés : Si nous souhaitons conserver la nature hiérarchique des données, nous devrions utiliser des modèles capables de gérer de telles structures :

- **Réseaux de neurones récurrents (RNN) ou réseaux Long Short-Term Memory (LSTM)** : Ceux-ci peuvent travailler avec des séquences de données et peuvent être capables de gérer des listes de tuples directement après un certain prétraitement.
- **Modèles basés sur les arbres** : Les arbres de décision ou des méthodes d'ensemble comme les Random Forests et le Gradient Boosting peuvent parfois gérer des structures de données plus complexes après aplatissement.

Structure des données

Quelle input donner ?

Donner un input tabulaire à l'ordinateur.
Chaque colonne est un "feature" qui permet les prédictions.

1 - Tableau avec bursts ?

- Complexifie l'annotation syntaxique
- Division des bursts en sous-séquences
- Informations "globales" sur les bursts

2 - Division du texte final ?

- Perte de certaines pauses
- Annotations syntaxiques facilitées
- Perte de la structure des bursts

Structure des données

Enrichir les données

Pour chaque burst (et donc pour chaque sous-séquence de bursts), on obtenir les informations suivantes :

- Annotation POS (liste de POS)
- Annotation CHUNKS (liste de chunks)
- Occurrences globales (occurrences du mot dans un corpus de langue française, évalue la rareté de chaque mot)
- Occurrences in text (nombre d'occurrences du mot dans le texte)
- Fréquence relative (nombre d'occurrences du mot dans le texte sur le nombre total de mots dans le texte)

Nous n'avons pas pris en compte les fréquences au sein du corpus.

Structure des données

Ajout de 5 colonnes

n_burst	burst Start	burstDur	pauseDur	cycleDur	burstPct	pausePct	total_Acti	total_Char	final_Char	total_Dele	innerDel	posStart	posEnd	docLen	categ	charBurst	ratio	POS	Chunks	Frequencies	Frequencies_in_text	Relative_frequencies_in_text
2	941.303	8.67	5.7	14.37	0.6	0.4	45	45	45	0	0	0	45	46	P	Dans le cadre de la médecine traditionnelle,	1.52	ADP,DET,NOUN,ADP,DET,NOUN,ADJ	nmod	4548946,3170,7225478,4134008,6524,785	1,6,1,21,12,12,1	0.00234,0.01402,0.00234,0.04907,0.02804,0.02804,0.00234
3	955.671	7.3	8.87	16.17	0.45	0.55	31	31	31	0	0	45	76	77	P	nous voici face à un dillemme.	0.82	PRON,VERB,NOUN,ADP,DET,NOUN	nsubj,obl:arg	1275361,71674,33918,3534420,3752833	3,1,3,10,3,1	0.00701,0.00234,0.00701,0.02336,0.00701,0.00234
4	971.839	21.78	8.49	30.27	0.72	0.28	100	97	94	3	3	76	128	171	P	Beaucoup critiqué cette dernière s'avère parfois ene	2.57	PROPN,VERB,DET,ADJ,ADV	nsubj	384,537137,83464,43775	1,1,5,3,4	0.00234,0.00234,0.01168,0.00701,0.00935
4	971.839	21.78	8.49	30.27	0.72	0.28	100	97	94	3	3	128	127	171	P	⌫	2.57					
4	971.839	21.78	8.49	30.27	0.72	0.28	100	97	94	3	3	127	170	171	P	effet fort efficace sur certaines points,	2.57	NOUN,ADV,ADJ,ADP	nmod	30167,50753,4301,801937	4,1,3,4	0.00935,0.00234,0.00701,0.00935
4	971.839	21.78	8.49	30.27	0.72	0.28	100	97	94	3	3	170	169	171	P	⌫	2.57					
4	971.839	21.78	8.49	30.27	0.72	0.28	100	97	94	3	3	169	168	171	P	⌫	2.57					
4	971.839	21.78	8.49	30.27	0.72	0.28	100	97	94	3	3	168	170	171	P	.	2.57					
5	1002.111	21.42	2.03	23.45	0.91	0.09	120	120	120	0	0	170	290	291	P	En effet si l'ont y réfléchie plus longuement beaucoup de nos méthodes actuelles se sont basé sur les fond de cette médi	10.55	ADP,NOUN,SCONJ,PRON,VERB,ADV,ADV,ADV,ADP,DET,NOUN,ADJ,PRON,AUX,VERB,ADP,DET,ADP,DET	obl:mod,obj,obl:arg,obl:arg,nmod	30167,1212854,1130167,206,1038200,710,182039,7225478,13862 0,2955,353,741792,519751,2318,801937,2606014,7225478,537137	1,4,5,1,1,10,1,3,21,4,2,2,3,5,1,4,12,21,5	0.00234,0.00935,0.01168,0.00234,0.00234,0.02336,0.00234,0.00701,0.04907,0.00935,0.00467,0.00701,0.01168,0.00234,0.00935,0.02804,0.04907,0.01168

Informations manquantes ou faussées à cause de la nature des productions

Structure des données

Informations séquentielles

La pause prédite est la dernière du burst, il serait donc intéressant d'avoir des informations au niveau des touches pressées.
Ajout d'une colonnes "pauses" pour chaque séquence de chaque burst.

categ	charBurst	pauses
P	Dans le cadre de la méde	[0.152, 0.136, 0.064, 0.032, 0.056, 0.432, 0.072, 0.2, 0.09
P	nous voici face à un dille	[0.104, 0, 0.072, 0.048, 0.128, 0.016, 0, 0.232, 0.001, 0.16
P	Beaucoup critiqué cette	[0.04, 0.024, 0.264, 0.128, 0.032, 0.104, 0.056, 0, 0.128, 0
P	⌫	[0.04, 0.024, 0.264, 0.128, 0.032, 0.104, 0.056, 0, 0.128, 0
P	effet fort efficace sur ce	[0.04, 0.024, 0.264, 0.128, 0.032, 0.104, 0.056, 0, 0.128, 0
P	⌫	[0.04, 0.024, 0.264, 0.128, 0.032, 0.104, 0.056, 0, 0.128, 0
P	⌫	[0.04, 0.024, 0.264, 0.128, 0.032, 0.104, 0.056, 0, 0.128, 0
P	.	[0.04, 0.024, 0.264, 0.128, 0.032, 0.104, 0.056, 0, 0.128, 0
P	En effet si l'ont y réfléc	[0.2, 0.256, 0.056, 0.176, 0.16, 0.144, 0.08, 0.072, 0.184,
ER	⌫	[0.408, 0.432, 0.024, 0, 0.128, 0.224, 0.12, 0.04, 0.095, 0.
ER	ecine alternative	[0.408, 0.432, 0.024, 0, 0.128, 0.224, 0.12, 0.04, 0.095, 0.
P	, certaine	[0.399, 1.119, 0.104, 0.136, 0.096, 0.119, 0.024, 0, 0.08, 0
P	⌫	[0.399, 1.119, 0.104, 0.136, 0.096, 0.119, 0.024, 0, 0.08, 0
P	s remède sont encà	[0.399, 1.119, 0.104, 0.136, 0.096, 0.119, 0.024, 0, 0.08, 0
P	⌫	[0.399, 1.119, 0.104, 0.136, 0.096, 0.119, 0.024, 0, 0.08, 0
P	ore utilisés	[0.399, 1.119, 0.104, 0.136, 0.096, 0.119, 0.024, 0, 0.08, 0
P	à nos jours	[0.296, 0.336, 0.096, 0.096, 0.04, 0.176, 0.248, 0, 0.04, 0.
P	. C	[0.776, 0.768, 0.288, 0.304, 0.752]
P	⌫	[0.776, 0.768, 0.288, 0.304, 0.752]
R	s	[0.416, 0.28, 0.288, 0.24, 0.112, 0.08, 0.096, 0.232, 0.224]

Structure des données

Padding et Masking

Certaines lignes n'ont pas de POS, nous avons donc des nombres de features différents. Pour remédier à ça, nous divisions par exemple la liste des pauses d'une ligne en autant de colonnes que nous avons de pauses.

Nous faisons de même pour toutes les colonnes contenant des listes et finissons avec un nombre gigantesque de colonnes. Les colonnes vident sont remplies avec une valeur que nous demanderons au modèle d'ignorer lors de l'apprentissage.

n_burst	deletions	Charbust _1_1	Charburst _1_2	Charburst _2_1	POS_1	POS_2
1	0	1,33330	0,4555	0,4563	0,44	0,089
2	0	1,4563	-99999	-99999	0,65	-99999
3	1	1,46555	0,1532	-99999	-99999	-99999

Flattened structure

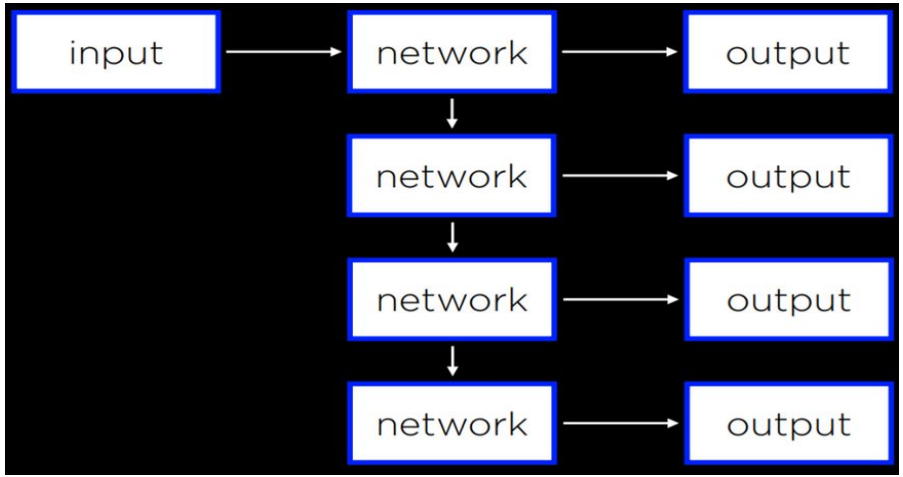
- En utilisant une régression linéaire, nous obtenons un score R^2 de -0.6802.

  Le coefficient R^2 détermine la qualité de notre régression. Quand ce dernier est positif, alors le modèle suit la tendance des données, plus il se rapproche de 1, plus le modèle est efficace. À l'inverse, s'il est négatif, alors le modèle ne suit pas du tout la tendance des données et les prédictions effectuées sont pires que si nous avions effectué une prédiction basée sur la moyenne. Plus il est bas, plus le modèle n'est pas efficace. Si le coefficient est de 0, alors le modèle prédit les valeurs aussi bien que la moyenne, il n'y a pas de valeur de prédiction !

- Donc, nous avons essayé d'autres modèles comme le Random Forest ou les arbres de décision. Cependant, nous n'avons pas réussi à obtenir de résultats satisfaisants.

Neural networks

- À cause des résultats peu convaincants, nous avons décidé d'utiliser un réseau de neurones et plus particulièrement un réseau de neurones récurrent. Nos données sont séquentielles, c'est à dire que nous traitons des séquences de données (des bursts) décrits par plusieurs features (durée, nombre d'effacements…).
- Nous choisissons un Recurrent Neural Network (RNN) car ce dernier est adapté aux données séquentielles. Un RNN traite chacune de nos séquences (bursts) en gardant en mémoire les précédentes et est ainsi adapté aux données ayant des relations temporelles.



Structure des données

Dans notre structure :

Séquence : Un burst composé de plusieurs lignes

Step : Une ligne d’un burst

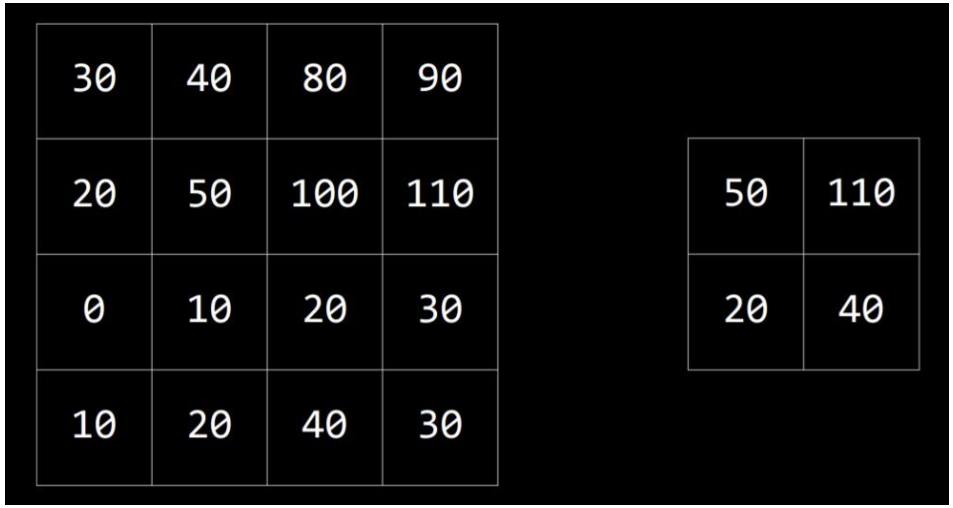
Chaque step a plusieurs features. Chaque feature apparaît dans une colonne. Certaines features décrivent le burst en entier (ex: nombre d’effacements) et d’autres décrivent une ligne (ex: catégorie R, P ou ER).

Chaque séquence doit avoir des steps de même longueur, il faut donc avoir le même nombre de colonnes pour chacun.

Essais

Recours obligatoire au pooling :

La taille de l'input est réduite en samplant à partir de certaines régions de l'input. Le pooling peut être fait en fonction de plusieurs stratégies : chercher les valeurs maximales de chaque régions ou celles les plus proches de la moyenne…



Nous avons choisi un pooling basé sur la moyenne (global pooling average).

Essais

Gérer les valeurs extrêmes

Certaines valeurs s’avèrent être extrêmement élevées. Certaines sont dues à des pauses correspondant à un enregistrement par exemple mais d’autres ne peuvent être expliquées.

Nous avons donc recours à une transformation logarithmique. La transformation logarithmique est une opération permettant de compresser les données et donc de donner moins d’importance aux données extrêmes, la variance est ainsi réduite.

Par exemple, log(100) et log(1000) sont proches (2 et 3) alors que 100 et 1000 sont éloignées (différence de 900).

Nous obtenons des résultats plus proches de 0 mais qui restent négatifs.

Essais

Reconnaître les valeurs pertinentes

Le RNN se focalise sur l'état final d'une séquence pour effectuer ses prédictions. Or, en faisant celà, des données précieuses de premiers steps peuvent être perdues.

Nous utilisons un attention mechanism, un processus qui permet aux modèles de prendre en compte différentes parties de l'input et ainsi retenir certaines informations pouvant positivement impacter les prédictions.

L'utilisation de l'attention mechanism améliore beaucoup les prédictions mais le R^2 reste négatif (et proche de 0…).

Essais

Éviter le surapprentissage

Au vue du nombre de données, il est important de ne pas tomber dans le surapprentissage.

Pour cela, nous utilisons l'hyper paramètre dropout rate qui permet de désactiver un nombre de neurones prédéfini à chaque exécution du modèle. Cela permet au modèle de ne pas se focaliser sur certaines attributs spécifiques et “importants” afin de pouvoir se focaliser sur le reste du set.

Les performances sont meilleures mais le R^2 reste négatif.

Essais

Obtention des hyperparamètres.

Obtenir les hyperparamètres permet d'avoir la combinaison optimales permettant d'avoir la meilleure performance. Le nombre de neurones utilisés, la dropout rate et la learning rate sont changées et combinées au sein de plusieurs essais.

Nous obtenons un R^2 toujours négatif avec les meilleurs hyperparamètres.

Une des limitations possibles serait le nombre de textes donnés en input à l'ordinateur, il est cependant impossible de données plus de 2 textes à l'ordinateur au risque de devoir attendre plusieurs heures.

On a essayé avec tous les textes et avons obtenu un temps d'exécution de 85h sur google collab. (sans oublier plusieurs crash !)