Rapport du projet TC3
Classification des oeuvres par genre et période

Issa Hammoud
Salim TABARANI

Master 2 AIC
Université Paris Saclays
Paris, France

16 novembre 2018

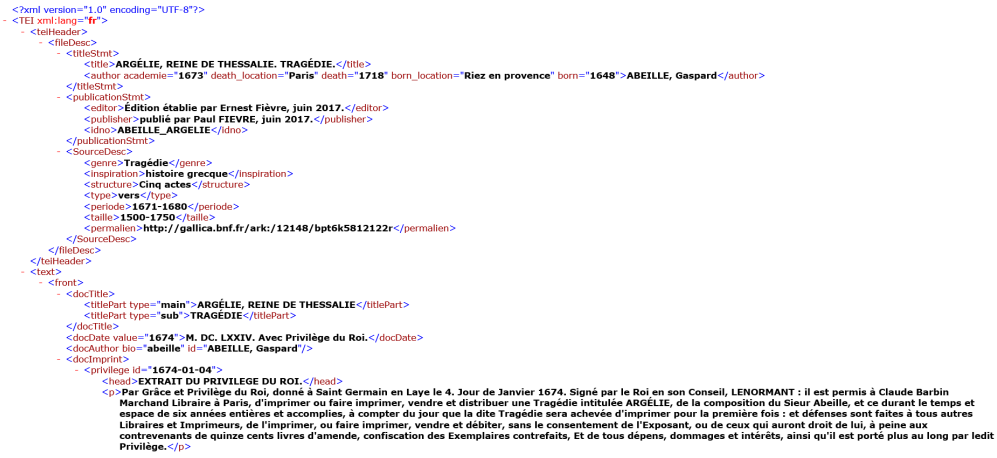
1 Introduction

Dans ce projet,nous utiliserons le corpus TheatreClassique afin d'établir un modèle capable de prédire les divers caractéristiques d'une pièce de theâtre à partir de son contenu.

2 Prétraitement des données

2.1 Nettoyage du dataset

Les données fournies se présentent sous format XML,Chaque oeuvre se divise en deux parties :Un header qui contient les caractéristques de la pièce comme le genre,l'inspiration,la structure et la période.Une partie text qui représente le contenu de la pièce.



Pour pouvoir exploiter le contenu de ces pièces,Une étape de nettoyage s'impose.En effet,les textes d'une pièce sont réparties en scènes at actes,donc nous les avions regroupé ensemble.
Ensuite,en utilisant le corpus "stopwords" de nltk,on enlevé les mots vides de ces textes pour diminuer leur taille et garder les éléments importants,ce qui permet également de gagner en efficacité.

2.2 Détermination de la composition du training set

Le corpus contient 1030 pièces.Ce qui ne constitue pas assez d'éléments pour l'entrainement et le test.Pour régler ce problème,nous avons séparé les actes de chaque pièces.Ce qui a donc permis d'obtenir un dataset caractérisé de la manière suivante :

-La taille du dataset est 3072 actes
-Le nombre moyenne des mots par actes (ponctuation incluse) est 2487
-Le nombre maximum des mots par actes (ponctuation incluse) est 17365
-Le nombre minimum des mots par actes (ponctuation incluse) est 75
-Le nombre de mots totales (non uniques) est 7848819
-Le nombre de mots totales (uniques) est 279086

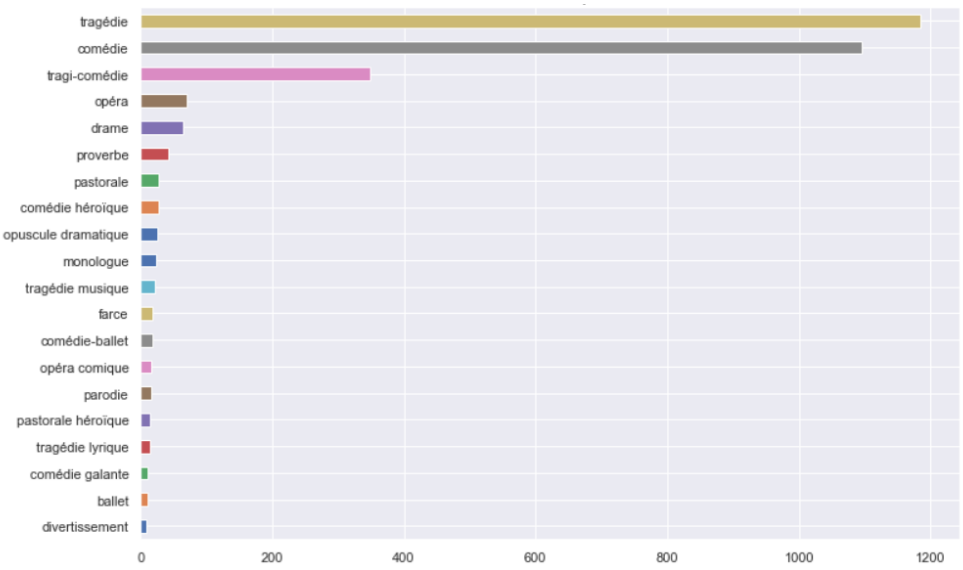
2.3 Détermination des caractéristiques à prédire

Maintenant que la structure des données a été établi et préparée pour l'apprentissage.Il reste à déterminer les caractéristiques intéressantes pour la prédiction.Nous avons décidé d'en choisir les deux suivantes :Genre et Périodes.
En effet,celles ci sont bien représentés et permettent de caractériser efficacement une pièce de theâtre.

3 Présentation du dataset

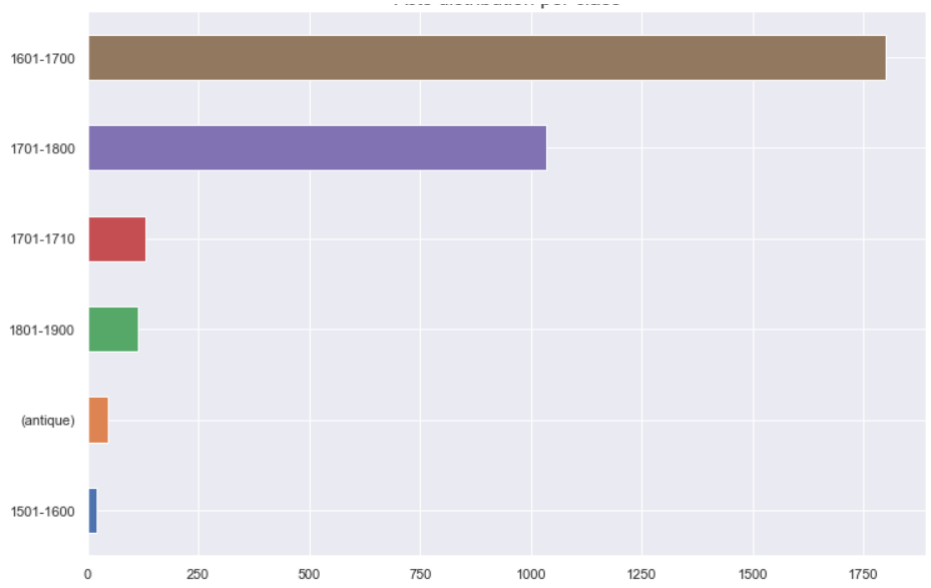
3.1 Genres

Sur l'ensemble des données,on observe 20 classes présentes au moins 10 fois.Cependant,on remarque que les classes :Tragédie,Comédie et Tragi-comédie sont beaucoupplus représentés.Ainsi,une première étude sera réalisé avec l'ensemble des classes.Ensuite,nous nous concentrerons uniquement sur les 3 classes les plus représentés.



3.2 Périodes

Les périodes sont indiquées dans les pièces sous la forme de tranche de 10 ans.Ce qui permet d'avoir un ensemble de 24 classes sur l'ensemble des donnés.Ce qui très grand et une réduction de ce nombre s'avère nécessaire.Donc nous avons choisi de catégoriser les périodes par siècles.Ceci permet d'obtenir 6 classes dont la répartition est représentée ci dessous :



4 Apprentissage

Comme tous les modèles d'apprentissage profond, on peut pas utiliser les données brutes telles quelles, il faut les transformer en données numériques, ce qui est connu sous le nom de "vectorization". Dans ce qui suit, on va parler de la méthode de vectorization utilisée dans ce projet, qui est le plongement de mots ou plus connu par son nom anglais "Word embeddings".

4.1 Choix de la méthode

C'est un moyen populaire et puissant pour associer un vecteur dense à un mot. L'avantage de cette méthode est que les vecteurs sont dans un espace de petite dimension. Ces vecteurs sont appris par les données alors on espère que dans cet espace la distance entre les mots corrélés est petite et celle entre les mots non corrélés est grande. On a donc utilisé 2 méthodes pour obtenir le word embedding :

-Les apprendre en même temps avec les autres paramètres du modèle. Dans ce cas, on les initialise aléatoirement et on les adapte comme les poids d'un réseau de neurone.

-Utiliser des word embedding déjà calculé sur autre corpus de données. Ils sont nommés word embeddings pré-entrainé.

4.2 Implémentation

Tout d'abord, il faut préciser la taille maximale du vecteur d'entrée. Puisque le nombre moyen de mots dans un acte est à peu près 2500, alors on l'a choisit. Dans ce cas, les vecteurs qui correspondent aux actes de moins de 2500 mots, vont être complété par des zéros.

Ensuite, il faut préciser la dimension de l'embeddings, c-a-d la dimension du nouvel espace. Mais puisqu'on va utiliser des embeddings pré-entrainé alors cette dimension est imposée à 200 pour notre choix de corpus.

Enfin, il faut indiquer le nombre maximal de mots, et puisqu'on a dans nos données a peu près 200000+ de mots uniques, alors on va choisir 50000 mots.

max_word	max_len	embedding_dim
100000	2500	200

On suivra donc la démarche suivantes :

-Découper le texte en maxword mots le plus commun avec Tokenizer.

-Transformer le texte en séquence.

-Mettre la taille à maxlen, ajouter des zéro si moins que maxlen et couper si plus.

-Définir un modèle séquentiel en Keras.

-Définir une couche embedding de taille (maxwords, embeddingdim, maxlen)

-Transformer cette couche en une couche 2D de taille (maxwords, embeddingdim * maxlen)

-Ajouter une couche cachée au dessus.

-Prédir avec softmax.

Pour les embeddings pré-entrainés, nous avons utilisé le corpus "frWaC". C'est un corpus de 1.6 millions de mots par le web :

-Utiliser les embeddings pré-entrainés pour créer une matrice de dimension (maxwords, embeddingdim)

-Insérer la matrice dans la couche embedding déjà créee.

-Préciser à Keras de ne pas entrainer cette couche.[2]

5 Résultats

Sans pré-entrainées	genre à 20 classes	genre à 3 classes	période à 6 classes
Résultat	71%	85.6%	84.9%

Avec pré-entrainées	genre à 20 classes	genre à 3 classes	période à 6 classes
Résultat	57%	68.4%	59.8%

En général, l'utilisation des embeddings pré-entrainés quand il n'y a pas beaucoup de données améliore les résultats. En revanche, nos résultats sont meilleurs sans utiliser ces embeddings pré-entrainés.

Ceci est dû au fait que l'embedding pré-entrainé utilisé contient juste 50000 mots non vides, ce qui est 5 fois moins que les mots de notre corpus.

6 Améliorations

C'est mieux d'utiliser un meilleur embedding pré-entrainé qui contient plus de mots, et qui sont lemmatisé par exemple. De plus, les méthodes des réseaux récurrents comme RNN et LSTM peuvent améliorer de plus la classification, alors c'est mieux d'utiliser juste un reseau de neurone normal au dessus de la couche embedding.

7 Bibliographie

-[1]http ://wacky.sslmit.unibo.it/doku.php ?id=corpora
-[2]Deep Learning with Python, Francois Chollet, 2018