M2 - Économétrie Appliquée et Statistiques

IAE Nantes - Université de Nantes

Projet SVM et réseaux de neurones : Wikipedia - Image/Caption Matching

Quentin CHIFFOLEAU

Loïc CORVEN

2021-2022

Sommaire

I - Introduction

Wikipédia est la plus grande encyclopédie collaborative disponible sur le web. Créée par Jimmy Wales et Larry Sanger le 15 janvier 2001. Il s'agit d'une œuvre libre, en accès gratuit où n'importe qui peut diffuser des informations sous forme d'article. En quelques années, elle est devenue l'encyclopédie la plus fournie et la plus consultée au monde. Wikipédia est disponible dans 300 langues, celle la plus utilisée, l'anglais, compte plus de 6 millions d'articles en début de 2020. Ces articles sont souvent agrémentés d'images comme support du texte.

C'est ici que se trouve l'objectif de notre projet Kaggle. En effet, le but est de prédire le titre ainsi que la description de la référence d'un dataframe composé d'images. Les données de ce projet sont scindées en deux fichiers : un dossier contenant les données d'apprentissage et un deuxième dossier contenant les données de test. S'agissant d'images, celles-ci prennent une place conséquente de plus de 300 Go.

Pour résoudre ce problème, nous utiliserons des techniques de machine Learning. Il s'agit de techniques d'apprentissage des machines à l'aide de données d'apprentissages. Ainsi des modèles sont créés à partir de cet échantillon d'apprentissage et permettent de prédire des caractéristiques sur des échantillons test. Pour que cela fonctionne au mieux, il faut donc disposer d'un maximum de données d'apprentissages et que la diversité y soit très forte pour que le modèle soit le plus efficace face à un maximum de situation. Nous utiliserons ici des modèles d'apprentissage supervisé.

Dans une première partie, nous procéderons à une analyse descriptive de nos données, puis à la préparation de celles-ci pour l'application de nos modèles. Nous vous présenterons ensuite les méthodologies que nous avons utilisées pour répondre à la problématique. Dans un dernier temps, nous analyserons nos résultats et concluons sur leurs pertinences.

II – Présentation et préparation des données

Notre base se compose de deux dossiers : un d'apprentissage et un de test. En effet, les données d'image d'apprentissage étant tellement volumineuse, elles ont dû être séparées du reste des fichiers. Dans les dossiers image_data_test et image_data_train se trouvent les données images : l'url de l'image originale, image encodée en base 64 avec une résolution de 300 pixels et les urls des pages communes où l'on peut retrouver cette image.

Dans le dossier test, nous disposons des fichiers listés ci-dessous :

```
: files_test = os.listdir(folder_test)
  for names in files_test :
      print(names)
```

```
image_data_test
sample_submission.csv
test.tsv
test_caption_list.csv
train-00000-of-00005.tsv
train-00001-of-00005.tsv
train-00002-of-00005.tsv
train-00003-of-00005.tsv
train-00004-of-00005.tsv
```

Sample_submission correspond au fichier où nos résultats de modèles seront rapatriés. Test correspond à la liste des urls d'images, chacune associée à un id auquel nous devons associer un titre et une référence de description présent dans le second fichier test_caption_list. Finalement, nous nous intéresserons aux différents fichiers train qui disposent de plus d'informations.

Ce fichier contient 18 colonnes que sont les suivantes :

- Language : la langue utilisée pour l'image
- Page_url : l'url de la page où se trouve l'image
- Image_url : l'url de l'image en question
- Page_title : le titre de la page
- Section_title : la section où se trouve la page
- Mime_type : le type d'extension pour l'image
- Original_height : la hauteur de l'image en pixels
- Original_width : la largeur de l'image en pixels
- Is_main_image : binaire pour savoir s'il s'agit de l'image principale de la page

- Attribution_passes_lang_id : binaire pour savoir si la langue de la page correspond à la description de l'image
- Page Changed Recently : binaire pour savoir si la page a été modifiée récemment
- Caption_title_and_reference_description : notre variable à définir pour test

Il y a également 6 autres variables, qui correspondent à des références de pages, de sections et sous-sections. Nous regardons ensuite le type de l'ensemble de ces variables :

```
: print(train.dtypes)
```

```
language                                  object
page_url                                  object
image_url                                 object
page_title                                object
section_title                             object
hierarchical_section_title                object
caption_reference_description             object
caption_attribution_description           object
caption_alt_text_description              object
mime_type                                 object
original_height                            int64
original_width                             int64
is_main_image                               bool
attribution_passes_lang_id                  bool
page_changed_recently                       bool
context_page_description                  object
context_section_description               object
caption_title_and_reference_description   object
dtype: object
```

Comme montré ci-dessus, la plupart des variables sont des chaînes de caractères, seuls la hauteur et la largeur de l'image sont des variables quantitatives et nous avons également trois variables binaires.

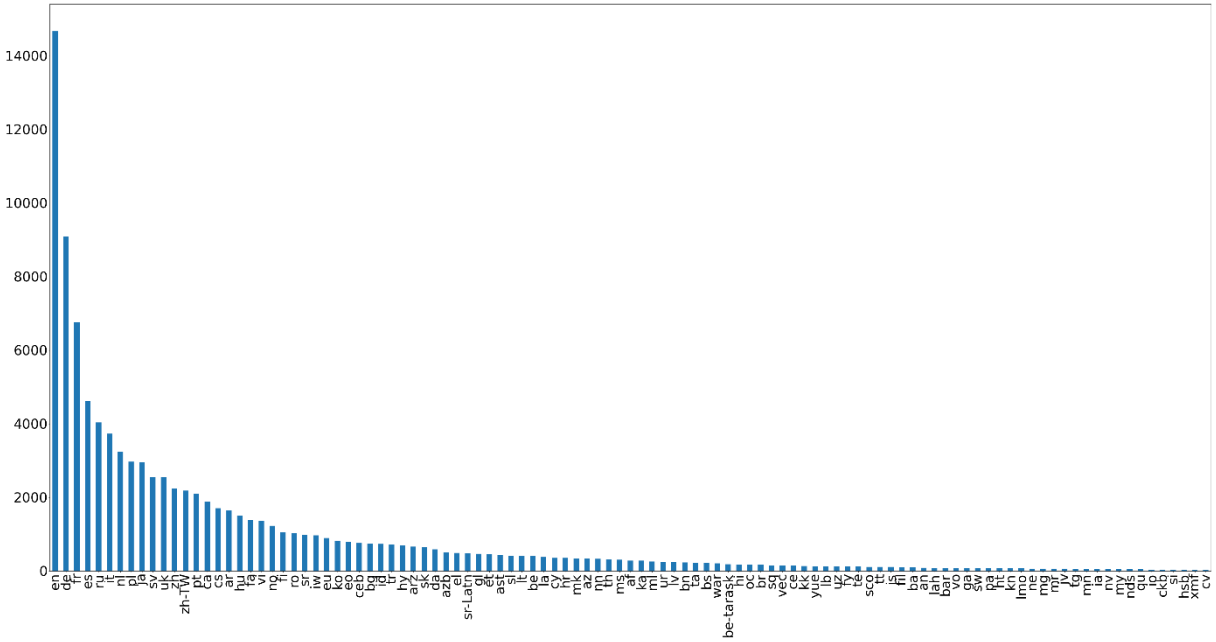
Nous regardons ensuite si nos données contiennent des valeurs nulles ou absentes à l'aide de la fonction .isnull(). Le test nous indique la présence de valeurs absentes. Nous regardons ensuite dans lesquelles se trouvent ces valeurs manquantes :

```
: train.isna().sum()/train.shape[0]*100
```

```
language                                   0.407
page_url                                   0.000
image_url                                  0.000
page_title                                 0.000
section_title                             51.996
hierarchical_section_title                 0.003
caption_reference_description             54.596
caption_attribution_description            6.136
caption_alt_text_description              85.501
mime_type                                  0.000
original_height                            0.000
original_width                             0.000
is_main_image                              0.000
attribution_passes_lang_id                 0.000
page_changed_recently                      0.000
context_page_description                   0.103
context_section_description               16.156
caption_title_and_reference_description    0.000
dtype: float64
```

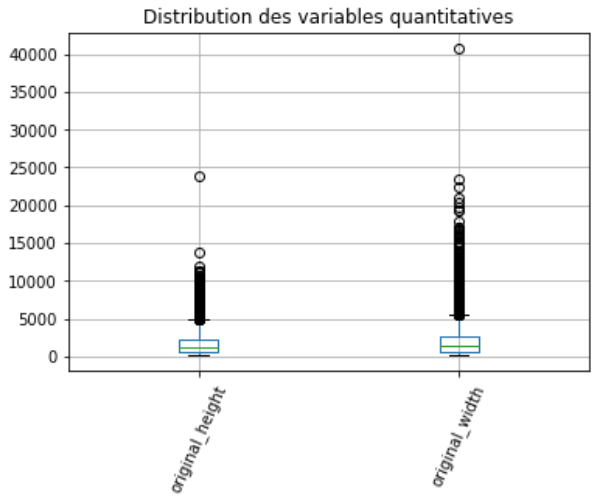
Comme indiqué ci-dessus en pourcentage du nombre d'observations, nous avons 8 variables qui contiennent des valeurs manquantes. Celles en contenant le plus sont les variables hierarchical_section_title, caption_reference_description, caption_alt_text_description et context_section_description avec respectivement 51.99, 54.59, 85.50 et 16.15 % de valeurs manquantes.

Afin d'avoir une idée des langues présentes, nous fabriquons un graphique en bâtons en fonction de la langue utilisée par chaque page :



Nous remarquons ainsi que le nombre de langues est très élevé avec l'anglais qui prédomine sur l'ensemble des autres. Notons que les 6 langues les plus présentes sont : l'anglais, l'allemand, le français, l'espagnol, le russe et l'italien.

Finalement nous procédons à la création de boxplots pour nos deux variables quantitatives :



On remarque que les images ont globalement la même taille. Cependant, nous supposons la présence d'outliers dans la partie haute de chaque boxplot. Cela paraît logique, signifiant que la plupart des images ont la même résolution et que seulement un nombre faible dispose d'une résolution élevée (hauteur et largeur élevées).

III – Application et modélisation

1) Prévision sur les URL

En regardant les données d'entraînement, on remarque un lien entre l'URL de la page et ce qu'il d'écrit. On remarque de même la même similitude avec l'URL de l'image qui semble contenir des informations sur sa description. De fait, nous avons également des URL dans les données tests. Nous allons donc dans cette première modélisation exploiter les URL pour faire des prédictions sur les légendes des images, uniquement basées sur les URL, sans regarder à proprement parler l'image. Nous entraîneront ainsi notre modèle sur les données d'entraînements, puis nous appliquerons ce modèle sur les données de test afin de juger de la qualité de ce dernier.

Tableaux 1: Les variables dans pour les données d'entrainements :

	language	page_url	image_url	page_title	section_title	hierarchical_section_title	caption_reference_description
0	fr	https://fr.wikipedia.org/wiki/Pariser_Kanonen	https://upload.wikimedia.org/wikipedia/commons...	Pariser Kanonen	Bilan	Pariser Kanonen / Bilan	NaN
1	tg	https://tg.wikipedia.org/wiki/Republic_P-43_La...	http://upload.wikimedia.org/wikipedia/commons/...	Republic P-43 Lancer	NaN	Republic P-43 Lancer	NaN
2	en	https://en.wikipedia.org/wiki/Deer_Park,_Wisco...	https://upload.wikimedia.org/wikipedia/commons...	Deer Park, Wisconsin	NaN	Deer Park, Wisconsin	Downtown Deer Park
3	pt	https://pt.wikipedia.org/wiki/Vaux-l%C3%A8s-Mo...	https://upload.wikimedia.org/wikipedia/commons...	Vaux-lès-Mouzon	NaN	Vaux-lès-Mouzon	NaN
4	ne	https://ne.wikipedia.org/wiki/%E0%A4%95%E0%A5%...	https://upload.wikimedia.org/wikipedia/commons...	क्रिसमस टापु	NaN	क्रिसमस टापु	NaN

Ce modèle se déroule en deux étapes. La première est d'extraire l'information descriptive contenue dans l'URL. Le package "urllib.parse" va décomposer les chaînes de caractères URL en extrayant les schémas d'adressages, l'emplacement réseau, chemin, etc. . De plus, la fonction unquote va nettoyer la partie information que l'on cherche à récolter afin d'obtenir de l'URL image une description de celle-ci. La description ainsi obtenue de l'URL sera notre prédiction.

Dans un second temps, nous avons utilisé la librairie "rapidfuzz" qui permet de calculer une distance (similarité) entre deux chaînes de caractères. Plus particulièrement, nous allons faire appel à la fonction "process.extract" de ce package. Cette dernière trouve les meilleures correspondances dans une liste de choix entre deux chaînes de caractères. Ainsi on va comparer la proximité de nos prévisions avec les données tests.

Après soumission de notre modèle sur KAGGLE nous obtenons un score suivant :

Name	Submitted	Wait time	Execution time	Score
submission2222.csv	just now	1 seconds	6 seconds	0.18064

Limite du modèle :

Si ce modèle est simpliste et est facile à mettre en place, il fait l'hypothèse forte que l'URL contient les informations nécessaires pour pouvoir décrire les images. Or dans le cas où cette hypothèse n'est pas respectée, les prévisions du modèle ne sont plus pertinentes. Nous allons développer un nouveau modèle pour pouvoir faire des prévision pertinentes non dépendantes de l'URL.

2) Prévision sur les images à partir d'une modélisation par réseau de neurones.

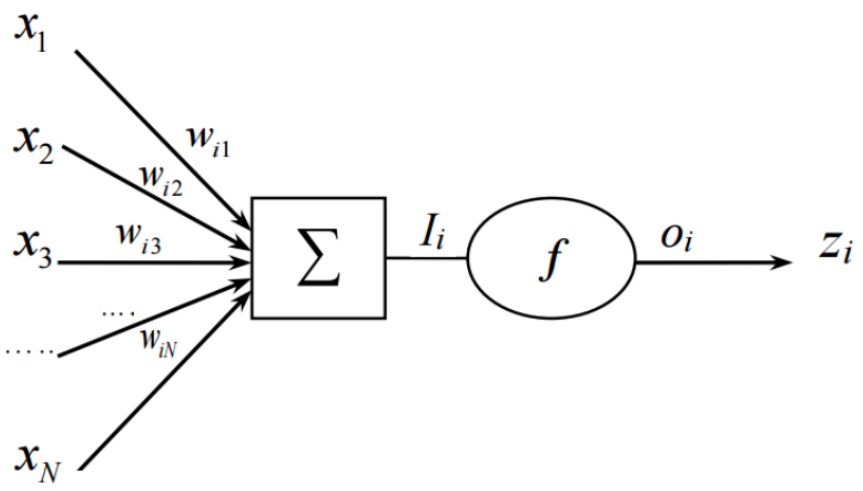
2.1 Introduction au réseaux de neurones convolutifs (CNN) :

Les réseaux de neurones convolutifs ont une approche similaire à celle des méthodes traditionnelles d'apprentissages supervisés. En effet, les neurones reçoivent des images en entrée, détectent les caractéristiques de chacune d'entre elles, puis entraînent un classifieur sur ces dernières. Là où les réseaux de neurones convolutifs se démarquent, c'est qu'ils apprennent les caractéristiques automatiquement. De même, contrairement aux techniques d'apprentissages supervisés, les réseaux de neurones convolutifs apprennent les caractéristiques de chaque image. Nous avons décidé d'appliquer les CNN sur notre problématique car c'est l'une des méthodes les plus appropriées dans le traitement et l'analyse des images.

Comment fonctionnent les réseaux de neurones convolutifs ?

les CNN simulent l'intelligence humaine au travers de neurones artificiels. Un neurone artificiel est un ensemble d'opérations mathématiques. Tout d'abord un poids et un biais sont appliqués à une valeur d'entrée. Dans le cas de notre problématique sur une analyse d'images, celle-ci est la valeur d'un pixel. Puis, une fonction d'activation est appliquée au résultat intermédiaire pour représenter les données dans l'espace des données de cette fonction. De fait, cette fonction d'activation est non linéaire dans la plupart du temps, car elle permet de représenter des données complexes où la combinaison linéaire ne fonctionne pas.

Schéma d'un neurone dans un réseaux :



L'architecture d'un réseau de neurones convolutifs est formée par une succession de blocs de traitement pour extraire les caractéristiques discriminant la classe d'appartenance de l'image des autres. Un bloc de traitement se compose d'une à plusieurs couches :

- Une couche de convolution qui traite les données d'un champ de récepteur.
- Une couche de correction qui dépend de la fonction d'activation.
- Une couche de pooling qui compresse l'information.

Comment fonctionne une convolution ?

Quand l'image arrive dans le réseau de neurones, l'image est découpée en sous-régions et est analysée par un noyau de convolution. L'analyse des caractéristiques de l'image par le noyau de convolution est une opération de filtrage avec une association de poids à chaque pixel. L'application du filtre à l'image est appelée une convolution. Les phases d'estimation et de validation du modèle sont répétées plusieurs fois jusqu'à l'obtention de bonnes performances. Ces itérations se base sur la méthode d'optimisation mathématique de rétropropagation et la descente de gradient.

Les CNN prennent en compte des hyper-paramètres à définir en amont de la modélisation. Ces derniers sont liés à la topologie et à la taille du réseau de neurones. Ainsi la taille du lot de données en entrée du réseau peut être définie (batch size), comme la méthode de chargement de données (data loader) qui peut-être considéré avec ou sans sous-échantillonnage. Nous retrouvons également des paramètres qui peuvent être définis

dans la phase d'entraînement et de validation. C'est le cas du nombre d'itérations (epoch) qui définit le nombre de boucles d'apprentissages que l'algorithme va réaliser pour améliorer les estimations de l'algorithme. La fonction de perte qui calcule la valeur de l'erreur existant entre l'estimation et l'observation. On retrouve également la fonction d'optimisation qui va influencer la descente de gradient. Et enfin le taux d'apprentissage (learning rate), qui est le pas de la descente de gradient.

2.2 Modélisation par CNN :

Bibliothèque utilisé :

PyTorch :

Bibliothèque open source développée par facebook, pytorch permet d'effectuer les caculculs tensoriels[1] nécessaires pour l'apprentissage profond (deep learning). Ainsi Pytorch va nous permettre de manipuler des tenseurs, ainsi que de calculer des gradients pour appliquer facilement des algorithmes d'optimisation par descente de gradient.

Weights & Biases :

Weights & Biase est une ensemble d'outils appliqués à l'apprentissage automatique. Deux outils sont particulièrement utilisés : les tableaux de bord qui vont permettre d'enregistrer et de visualiser les expériences en temps réel. Ainsi que les artefacts, qui sont un ensemble de données et de versionnage de modèle.

[1] tableaux multidimensionnels

La première difficulté avant de créer la modélisation est de préparer les données pour pouvoir utiliser pytorch. Cela a été effectué grâce à la classe dataset fourni par pytorch. Mais pourquoi préparer les données ? La structure des données est de type Numpy. Or cette structure ne peut pas être directement introduite dans le réseau car PyTorch nécessite l'utilisation des données en entrée du type tenseurs.

Nous avons donc utilisé en premier lieu un module Dataset pour créer une class personnalisée et initialisé l'ensemble des données. La chose importante à noter dans cette class est la fonction torch.tensor qui va permettre de convertir les données d'entrainements en tenseurs. De plus les images sont en base64 , il faut donc les convertir en format RGB.

Une fois les données préparées grâce à la class que nous avons appelé "WikipediaDataset" ; nous les chargeons en lot grâce à la fonction DataLoader. Les données sont actuellement prêtes pour l'entraînement des réseaux de neurones. Nous avons divisé ces données en deux parties pour l'apprentissage et le test. De fait, nous avons utilisé un échantillon de 500 images pour tester le modèle dans un premier temps. Nous allons ainsi pouvoir concevoir le réseau neuronal. Nous avons utilisé la fonction nn.Module pour créer ce dernier.

Les paramètres utilisées pour le réseau de neurone sont les suivants :

```
CONFIG = {"seed": 2021,
          "epochs": 3,
          "img_size": 256,
          "image_model_name": "tf_efficientnet_b0",
          "text_model_name": "xlm-roberta-base",
          "embedding_size": 256,
          "train_batch_size": 10,
          "valid_batch_size": 10,
          "learning_rate": 0.10,
          "scheduler": 'CosineAnnealingLR',
          "min_lr": 0.50,
          "T_max": 50,
          "weight_decay": 0.10,
          "max_length": 32,
          "n_accumulate": 1,
          "device": torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
          }
```

De plus, une fonction de pertes a été mise en place. Elle va évaluer l'écart entre les prédictions réalisées par le réseau de neurones et les valeurs réelles des observations utilisées pendant l'apprentissage. De fait, plus le résultat de cette fonction est minimisé (l'écart entre la valeur prédite et la valeur réelle) , plus le réseau de neurones est performant. Le modèle comprend également une fonction d'entraînement sur les données d'entraînement et une fonction test permettant de tester le modèle pour les données test.

Limite du modèle :

Malheureusement, après avoir fait tourner le modèle pendant plus de 10H, nous n'obtenons aucun résultat (le modèle tourne toujours) et cela même en réduisant la base drastiquement à 50 images, avec 25 dans dans la base d'entraînement et 25 dans la la base de validation. Nous ne pouvons ainsi pas conclure sur ce modèle. Nous pouvons émettre l'hypothèse suivante pour expliquer la défaillance de notre modèle : les ressources demandées par le modèle sont beaucoup trop importantes comparativement à la capacité de nos ordinateurs.

IV - Conclusion

Le modèle de prévision sur URL offre des premiers résultats exploitables. Ce dernier est simple à mettre en place et demande peu de ressources informatiques pour son exploitation. Cependant ces prévisions n'apparaissent pas comme optimums. Le modèle par CNN est celui qui offre en théorie les meilleurs résultats, cependant nous n'arrivons pas à le mettre en place.

V - Sitographie

https://docs-test.wandb.ai/ref/python/init

https://pytorch.org/

https://docs.python.org/3/library/urllib.parse.html

https://docs.python.org/3/library/urllib.parse.html

https://docs.wandb.ai/