Rapport du projet de classification d'un corpus chinois

Résumé

Ce projet a pour but de réaliser une classification sur un ensemble de commentaires à propos des produits quotidiens en 10 catégories. En souhaitant réaliser et comparer plusieurs algorithmes avec différentes représentations des données, j'ai donc réalisé plusieurs scripts pythons qui permettent de classifier ces commentaires à l'aide du machine learning. Les meilleurs résultats sont 89% de f-mesure avec la méthode Naive Beyes avec la représentation de Bow (mot-de-sac) et unigrammes, 89% de f-mesure avec la méthode Régression logistique avec la représentation de tf-idf et unigrammes et 89% avec LSTM (Long Short-Term Memory).

Introduction

Le corpus choisi sont toutes des commentaires chinois des utilisateurs provenant de sites web de commerce électronique. La tâche réalisée est celle de la classification multi-classe d'un ensemble de 60 milles commentaires et chaque commentaire ne peut correspondre qu'à l'une des 10 catégories. J'ai donc un premier temps testé la méthode de la Naïve Bayes, et puis la méthode de la Régression Logistique en utilisant la bibliothèque Sklearn. Ensuite, une autre version en utilisant LSTM a été adopté à l'aide de la librairie Keras. Dans la suite de ce rapport je présenterai le corpus et les prétraitements opérés, la méthodologie, l'implémentation et les résultats et enfin, les difficultés.

Description de données

Le corpus[1] est un ensemble de commentaires en chinois sur les équipements de la maison. Il est composé de 62,774 commentaires comportant au total environ 220,953 mots avant le nettoyage. C'est fichier csv avec un commentaire par ligne. Il est séparé en 2 colonnes : le commentaire(« review »), et la catégorie (« cat »). Les dix catégories sont 书籍(livres), 平板(tablettes), 手机(téléphones portables), 水果(fruits), 洗发水(shampoing), 热水器(chauffe-eau), 衣服(vêtements), 计算机(ordinateurs), 酒店(hôtels) et 蒙牛(Mengniu lait). Voici un exemple du corpus avec du code python :

```
In [5]: import pandas as pd
        df = pd.read_csv('corpus.csv')
        df=df[['cat','review']]
        print("Total number: %d ." % len(df))
        df.sample(10)
```

```
Total number: 62774 .
```

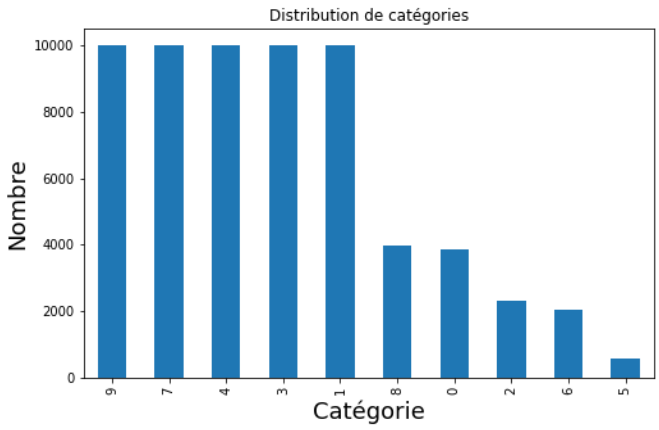
Out[5]:

	cat	review
20876	水果	苹果不错 水很多 拿到后第一时间就榨汁了！不过个头稍微小了点！！！
57594	酒店	房间相当大，干净整洁。周围有很多学校住宅小吃店，比较安静，小巴知道酒店门口，交通也相当方便，...
27747	洗发水	活动时买的比较超值，次日达，包装精美。
17050	水果	习惯好评，还没吃，吃了，再来追评！
19789	水果	苹果吃起来很甜，当做早餐吃的，不错去取货的时候没有在冷冻包里面，有点耽耽于怀！
54979	酒店	6月30日入住的。房间总体还行，就是有点旧。周围环境较好。服务方面，门童帮忙开车门倒是挺勤快...
61661	酒店	设施设备太老了，老四星酒店了，地理位置还行，隔音太差，大半夜能听到那种声音，总体来说，很一般
15194	手机	1、拍照片太黄；2、对焦不清晰，模糊；3、最大分辨率不能变焦；4、客服太烂。
15694	手机	1.查电话号码不够人性化,要逐个查看,或要输入名称查,晕!2.输入状态要退回上级目录要按N次...
49304	计算机	以前在京东买过一个as5530的机器3999元。也是780g的主板，用起来还可以。这次给表妹...

(Image 1 : Présentation du corpus)

Avant de réaliser les différents prétraitements, j'ai donc remplacé ces catégories pour faciliter les traitements par les chiffres comme ci-dessous :

[1] Le source de corpus : https://github.com/mei233/ChineseNlpCorpus/blob/master/datasets/online_shopping_10_cats/intro.ipynb



Catégorie	Cat_id	Nombre de commentaire
书籍(livres)	0	3851
水果(fruits)	1	10000
手机(téléphones portables)	2	2323
洗发水(shampoing)	3	10000
酒店(hôtels)	4	10000
热水器(chauffe-eau)	5	575
蒙牛(Mengniu lait)	6	2033
衣服(vêtements)	7	10000
计算机(ordinateurs)	8	3992
平板(tablettes)	9	10000

(Tableau 1 : Présentation de la catégorie)

En comptant les nombres de chaque catégories, j'observe que les données est déséquilibré avec seulement 575 commentaires sur les chauffe-eaux, 3992 commentaires sur les ordinateurs et même dix mille commentaires sur les tablettes. Ainsi, cela pourrait poser un problème à traiter lorsque j'ai divisé les donnée en train et test dans l'étape de la préparation des données.

Prétraitements opérés

Les prétraitement des données contient 4 parties : la suppression des doublons, la suppression des stopwords et/ou les ponctuations si nécessaire, la segmentation des phrases, et le filtrage des commentaires,.

Contrairement aux langes latine dont les lemmes sont constitués de phonèmes assemblés en morphèmes, les caractères chinois sont fixés et le lemme se réduit à un lexème ou un assemblage de lemmes (mot composé), sans morphèmes. Par exemple, le verbe pour « manger » se dit 吃饭 (chīfàn) quand il est employé sans complément d'objet (littéralement : « je mange de la nourriture ») mais 吃 (chī) quand le complément d'objet est exprimé : 我吃鱼 (wǒ chī yú) « je mange [du] poisson. ». En plus, les mots ne changent jamais leurs formes et les verbes ne se conjuguent pas. Par conséquence, la segmentation des phrases chinoises joue un rôle très important dans la segmentation des sens et de la désambiguïsé.

Pour la suppression des doublons, j'ai effectué une ligne de code à l'aide de la librairie Pandas :

```
data = data.drop_duplicates(subset=['review'], keep='first', inplace=False)
```

La méthode « drop_duplicates » permet de supprimer les doublons et le paramètre « keep = 'first' » va prendre la premier et supprimer les restes répétitifs.

Ensuite, pour obtenir un corpus sans ponctuations, j'ai utilisé les expression régulières. Dans la figure ci-dessous, ceux qui dans la colonne « review » sont les textes originaux et ceux qui dans la colonne « clean_review » ont été supprimé les ponctuations. Cette étape n'est pas toujours nécessaire. On va comparer différentes combinaisons de prétraitement afin de trouver la meilleure.

review	cat_id	clean_review
我已经爱上你了 蒙牛优益C!超级喜欢的味道	6	我已经爱上你了蒙牛优益C超级喜欢的味道
今天刚收到未用 应该不错 先不枉下评价等用一段时间再给发表使用情况、心得体验给大家提供参考	8	今天刚收到未用应该不错先不枉下评价等用一段时间再给发表使用情况心得体验给大家提供参考
钢琴漆虽好看，但太容易留手印和摩花了。风扇总是一直转，有时温度并不高也转。惠普还是一贯的不带...	8	钢琴漆虽好看但太容易留手印和摩花了风扇总是一直转有时温度并不高也转惠普还是一贯的不带驱动盘装...
买了两箱，物流好快，苹果很新鲜，汁水很多，脆脆的，不错。	3	买了两箱物流好快苹果很新鲜汁水很多脆脆的不错
定的是12/8 一晚，付款花了31分钟。结果订单自动取消。我在33分钟内付款成功，系统却自动...	9	定的是128一晚付款花了31分钟结果订单自动取消我在33分钟内付款成功系统却自动默认了第二天...
知道6.18要降价，价格保护既然没用了。不合理差评！！	1	知道618要降价价格保护既然没用了不合理差评
蛮好吃的也很划算不错赞一个还会继续购买的	3	蛮好吃的也很划算不错赞一个还会继续购买的
跟看到的图片不一样，有一行小字写的不一样。还有上面的盖也和商场的不一样，商场的上面没字，这个...	4	跟看到的图片不一样有一行小字写的不一样还有上面的盖也和商场的不一样商场的上面没字这个有个开字...
手机和平板都是华为的，支持国货。	1	手机和平板都是华为的支持国货
酒店设施很差，根本不像四星酒店。卫生间的镜子边上已破烂不堪，马桶也下不去水。酒店里吃晚饭时，...	9	酒店设施很差根本不像四星酒店卫生间的镜子边上已破烂不堪马桶也下不去水酒店里吃晚饭时只有3桌但...

(Image 2 : Suppression des ponctuations)

Pour supprimer les stopwords, j'ai utilisé une liste de stowpords chinois. Et pour la segmentation des phrase chinoise, j'ai utilisé la librairie Jieba[2] qui basé sur une structure de dictionnaire de préfixes pour obtenir une analyse efficace du graphe de mots. J'ai segmenté les mots en fonction de « clean_review » et divisé chaque contenu en mots individuels séparés par une espace. Les résultats sont dans la colonne « cut_review ».

review	cat_id	clean_review	cut_review
京东快递员真的超赞，服务到位，我们家老弱孕，没办法下楼拿，快递小哥二话不说给我抬上来了，很感...	3	京东快递员真的超赞服务到位我们家老弱孕没办法下楼拿快递小哥二话不说给我抬上来了很感动很棒	京东 快递 员 真的 超赞 服务到位 家 老弱 孕 没 办法 下楼 快递 小哥 二话不说 抬...
宝贝已收到两天了，质量很好，这次够狠满意，安装是我自己安装的，很容易，卖家服务很令我满意，好评。	5	宝贝已收到两天了质量很好这次够狠满意安装是我自己安装的很容易卖家服务很令我满意好评	宝贝 收到 两天 质量 够 狠 满意 安装 安装 容易 卖家 服务 很令 满意 好评
实在不好，差，电话难打，服务不好，客人退房时，服务员检查房间，只检查宾馆自己的东西，客人的忘...	9	实在不好差电话难打服务不好客人退房时服务员检查房间只检查宾馆自己的东西客人的忘记在房间的东西...	实在 不好 差 电话 难 服务 不好 客人 退房 时 服务员 检查 房间 检查 宾馆 东西 ...
没有操作系统，内存2G就好了，散热不好，时间长了键盘区左半边，以及掌托和触控板的左半边，比较烫。	8	没有操作系统内存2G就好了散热不好时间长了键盘区左半边以及掌托和触控板的左半边比较烫	没有 操作系统 内存 2G 散热 不好 时间 长 键盘 区 左 半边 掌托 触控板 左 半边...
房间里味道真的很臭，wifi连不上	9	房间里味道真的很臭wifi连不上	房间 里 味道 真的 臭 wifi 连不上
地点非常好，购物方便。店内布置较温馨，早餐不错，房间干净，服务态度也好。缺点就是不隔音，房门...	9	地点非常好购物方便店内布置较温馨早餐不错房间干净服务态度也好缺点就是不隔音房门就跟没有一样楼...	地点 非常 购物 方便 店内 布置 温馨 早餐 不错 房间 干净 服务态度 缺点 不 隔音 ...
不怎么的，刚买两天死机，要求换货不换，二货客服	1	不怎么的刚买两天死机要求换货不换二货客服	不怎么 刚 买 两天 死机 要求 换货 不换 二货 客服
平板手机很大，系统流畅。玩游戏发热小，除了换卡要关机重启这点不是很好。	1	平板手机很大系统流畅玩游戏发热小除了换卡要关机重启这点不是很好	平板 手机 很大 系统 流畅 玩游戏 发热 换卡 关机 重启 这点 不是
怎么说呢，第一次在京东买生活用品，我 买 的 是 假货！假货！假货！ 太可恶了，根本就不是清...	4	怎么说呢第一次在京东买生活用品我买的是假货假货假货太可恶了根本就不是清扬我还是在315打假前...	说 第一次 京东 买 生活用品 买 假货 假货 假货 太 可恶 根本 不是 清扬 315 打...
收到了裤子，裤子质量超好，弟弟穿上好看，洗了不缩水不掉色。很喜欢	7	收到了裤子裤子质量超好弟弟穿上好看洗了不缩水不掉色很喜欢	收到 裤子 裤子 质量 超好 弟弟 穿 好看 洗 不缩水 不 掉色 喜欢

(Image 3 : Segmentation des phrases)

Puisque la distribution des données est pas équilibré et certains commentaires qui sont hors sujet et ils sont plutôt courts, j'ai effectué donc un filtrage des phrases de moins de 5 mots pour obtenir un corpus dont les commentaires sont plus concrets. Le nombre de phrases obtenu après le filtrage est 60,148.

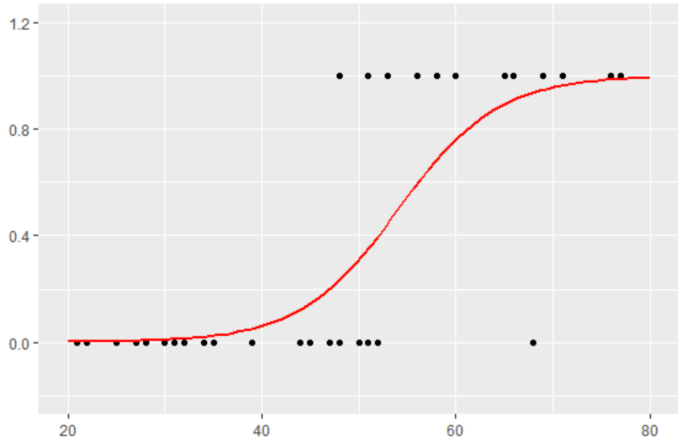
Méthodologie

Principalement, je vais comparer la méthode de la Naïve Bayes, la Régression Logistique et celle de LSTM. Les méthodes de la Naïve Bayes et de la Régression logistique fait partie de la Machine Learning traditionnel, et les réseaux de longue mémoire à court terme (LSTM) sont une extension pour les réseaux neuronaux récurrents, qui étend leur mémoire.

[2] La source de Jieba : https://github.com/fxsjy/jieba

La méthode de la Naïve Bayes est une collection d'algorithmes de classification basés sur le théorème de Bayes, à savoir que chaque caractéristique classée est indépendante de la valeur de toute autre caractéristique. Cependant, les caractéristiques ne sont pas toujours indépendantes, ce qui est souvent perçu comme un inconvénient de l'algorithme Naïve Bayes.

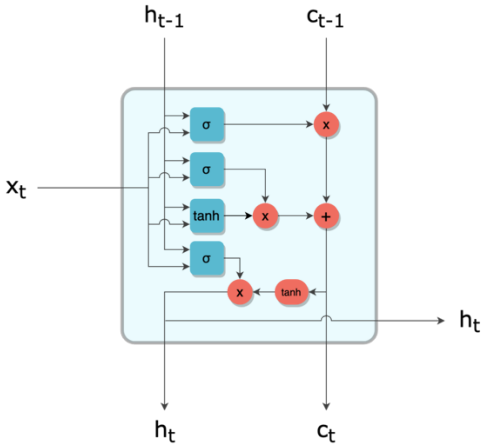
La régression logistique est un modèle statistique permettant d'étudier les relations entre un ensemble de variables qualitatives Xi et une variable qualitative Y. Il s'agit d'un modèle linéaire généralisé utilisant une fonction logistique comme fonction de lien. Ce n'est pas la réponse binaire qui est directement modélisée, mais la probabilité de réalisation d'une des deux modalités et cette probabilité est alors modélisée par une courbe sigmoïde, bornée par 0, et 1 :



(Image 4 : Fonction sigmoïde)

Lorsque la valeur prédite est supérieure à un seuil, l'événement est susceptible de se produire, alors que lorsque cette valeur est inférieure au même seuil, il ne l'est pas.

Quant aux LSTM, ils permettent aux RNN de se souvenir de leurs intrants sur une longue période de temps. C'est parce que les LSTM contiennent leurs informations dans une mémoire, ce qui ressemble beaucoup à la mémoire d'un ordinateur parce que le LSTM peut lire, écrire et supprimer des informations de sa mémoire. C'est aussi la raison pour laquelle que le LSTM est très utilisé pour les tâches de TAL.



(Image 5 : Représentation simplifiée d'une cellule LSTM - input, hiddden state, cell state)

Le fonctionnement global d'un LSTM peut se résumer en 3 étapes : ①Détecter les informations pertinentes venant du passé, piochées dans le cell state à travers la forget gate ; ② Choisir à partir de l'entrée courante, celles qui seront pertinentes à long terme, via l'input gate. Celles-ci seront ajoutées au cell state qui fait office de mémoire longue ; ③ Piocher dans le nouveau cell state les informations importantes à court terme pour générer le hidden state suivant à travers l'output gate. Et la couche sigmoïde σ génère des nombres compris entre 0 et

1, décrivant la quantité de chaque composant qui doit être laissée passer. Une valeur de 0 signifie « ne laissez rien passer », tandis qu'une valeur de 1 signifie « laissez tout passer! ». Le modèle peut augmenter ou diminuer les informations grâce à un tel traitement.

Implémentation et Résultats

Pour les méthodes de la Naïve Bayes et la Régression Logistique, j'ai utilisé la librairie scikit-learn, et la librairie keras pour le LSTM en langage python.

Pour ces deux méthodes du Machine Learning traditionnels, j'ai décidé de diviser les donnée 80 % pour le training set et 20 % pour le testing set et les sauvegardés en fichiers train_file.csv et test_file.csv avant d'entraîner un modèle. Donc j'ai réalisé un script python qui me permet de diviser les données et faire des prétraitements nécessaires en même temps. Afin d'obtenir les données plus équilibres, lorsque j'ai effectué la méthode `train_test_split()` de module `sklearn.model_selection`, j'ai utilisé l'argument `stratify = Corpus.cat` pour qu'ils soient divisés en fonction de ces 10 catégories.

Ensuite, j'ai testé différents combinaisons avec ces deux méthodes après avoir effectué le traitement de segmentation. Afin de réaliser la représentation Bow (Bag-of-words), j'ai utilisé la méthode `CountVectorizer()` de la librairie scikit-learn et pour la représentation tf-idf, j'ai appliqué la méthode `TfidfVectorizer()` dont l'argument `ngram_range` qui nous permet de générer N-gramme en même temps.

Principalement, j'ai effectué quatre prétraitements différents pour obtenir les représentations des données différentes et ensuite j'ai les implémenté avec la méthode de la Naïve Bayes et la méthode de la Régression Logistique respectivement.

Description des quatre prétraitements différents :

A : Suppression des caractères inutiles et des signes de ponctuations
B : Suppression des stopwords
C : Suppression des caractères inutiles, des ponctuations et les stopwords
D : Suppression les stopwords et filtrage des phrases de moins de 5 mots

Ci-dessous sont les résultats obtenus, pour chaque combinaison, j'ai donné également les meilleurs résultats contenant la f-mesure et le temps de l'entraînement du modèle.

		A	B	C	D
1	Naïve Bayes + BOW + Unigramme	F1 = **0.86** T = **1.6s**	F1 = **0.88** T = **2s**	F1 = **0.86** T = 1.8s	F1 = **0.89** T=1.7s
2	Naïve Bayes + BOW + Unigramme + Bigramme	F1 = 0.85 T=1.8s	F1 = 0.87 T = 4.9s	F1 = 0.85 T = 4.86s	F1 = 0.88 T = 4.9s
3	Naïve Bayes + TF-IDF +Unigramme	F1 = 0.85 T = 1.8s	F1 = 0.87 T = **1.7s**	F1 = 0.82 T = **1.7s**	F1 = 0.88 T = **1.6s**
4	Naïve Bayes + TF-IDF + Unigramme + Bigramme	F1 = 0.82 T = 5.7s	F1 = 0.87 T = 5.4s	F1 = 0.84 T = 5.2s	F1 = 0.86 T = 5s

(Tableau 2 : Résultats sur différentes combinaisons de la méthode Naïve Bayes)

En observant les résultats, on observe que la représentation de Bow et unigrammes avec la méthode de la Naïve Bayes est la plus performante pour tous les quatre prétraitements. En général, le traitement de la suppression de stopwords fonctionne légèrement mieux que celle de la suppression des caractères inutiles et des signes de ponctuations. En revanche, la suppression des caractères inutiles et des signes de ponctuations et de stopwords n'a pas produit un meilleur résultat. Après observation du corpus, j'ai découvert que de nombreux commentaires sur les chauffe-eaux qui racontent seulement le service de la livraison au lieu de la qualité du produit sont court et ne sont pas assez concrets telles que « (物流很快)la livraison était vite » et « (好评)c'est bon ». En conséquence, j'ai supprimé des phrases qui contient moins de 5 mots. Les résultats se sont améliorés et la représentation par Bow avec les unigrammes a même atteint 0,89 de la f-mesure.

Comparant avec les résultats de la méthode de la Régression Logistique ci-dessous, l'un des traits distinctifs est que la méthode de la Naïve Bayes sont beaucoup plus rapide que ceux de la Régression logistique. Le modèle le plus performant réalisé par la méthode de la Régression Logistique a pris 13 secondes, tandis que celui de la méthode de la Naïve bayes n'a pris que 1,7 secondes, ce qui est environ dix fois plus rapide.

		A	B	C	D
5	Régression Logistique + BOW + Unigramme	F1 = 0.86 T = **14s**	F1 = 0.87 T = **14s**	F1 = 0.87 T = 13s	F1 = 0.88 T=**13s**
6	Régression Logistique + BOW + Unigramme + Bigramme	F1 = 0.87 T=99s	F1 = 0.87 T = 92s	F1 = 0.87 T = 97s	F1 = 0.88 T = 90s
7	Régression Logistique + TF-IDF +Unigramme	F1 = 0.87 T = 18s	F1 = **0.88** T = 17s	F1 = 0.87 T = 18s	F1 = **0.89** T = 14s
8	Régression Logistique + TF-IDF + Unigramme + Bigramme	F1 = **0.88** T = 96s	F1 = 0.88 T = 98s	F1 = 0.86 T = 101s	F1 = 0.88 T = 74s

(Tableau 3 : Résultats sur différentes combinaisons de la méthode Régression Logistique)

Selon les résultats obtenus ci-dessus, on observe que les représentations de tf-idf sont probablement mieux de que celles de BOW avec la f-mesure légèrement plus élevée que celles de BOW. En plus, en ce qui concerne les prétraitements de corpus, la combinaison de la suppression des ponctuations et des stopwords ne produisent les meilleurs résultats, ce qui est identique à la méthode de la Naïve Bayes.

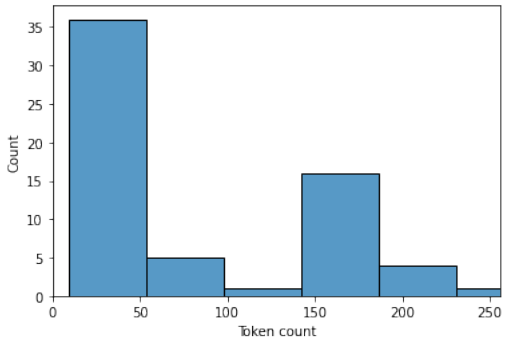
D'ailleurs, quant au temps de l'entraînement de modèle, la représentation des unigrammes a pris toujours moins de temps que les autres et la représentation TF-IDF et les unigrammes avec les prétraitements des stopwords et le filtrage des phrases de moins de 5 mots a pris seulement 13 secondes pour l'entraînement et a atteint 0.89 comme la f-mesure.

Néanmoins, la représentation de tf-idf avec les unigrammes et les bigrammes ont pris beaucoup plus de temps en raison de nombreux de données.

En conclusion, j'ai choisi la représentation tf-idf des unigrammes avec le prétraitements de la suppression des stopwords et le filtrage de moins de 5 mots pour la méthode Régression Logistique. C'est parce qu'il est premièrement plus performant que ceux de la représentation BOW avec la f-mesure de 0.88 et qu'il est beaucoup plus efficace que ceux de la représentation tf-idf avec les unigrammes et les bigrammes avec seulement 14 secondes de temps d'apprentissage.

Pour l'entrainement d'un modèle LSTM, j'ai d'abord pris des prétraitements nécessaires et la segmentation grâce à la librairie Jieba. A l'aide de la classe `Tokenizer` de la librairie `Keras`, j'ai réussi à vectoriser les données textuelles segmentée en transformant chaque texte soit en une séquence d'entiers avec les méthodes `fit_on_texts()` et `texts_to_sequences()`.

Ensuite, on devrait « padding » chaque chaîne pour qu'il aient la même taille avec la méthode `pad_sequences()`. Donc, pour obtenir la taille maximum, j'ai calculé la distribution de nombre de mot par commentaire. On pourrait observer que la taille maximum est environ 250 selon la graphique présentant la distribution des nombres de mots par phrase ci-dessous.



(Image 6 : Distribution des nombres de mot par commentaire)

Contrairement aux deux autres méthodes, j'ai divisé les donnée 90 % pour le training set et 10 % pour le testing set parce que les modèles RNN sont gourmand des données. Ensuite, j'avais défini premièrement la structure du modèle en ajoutant une couche « Embedding » de 5000 maximum de mots, 100 dimensions et 250 pour taille d'entrée comme défini précédemment. Puis, j'ai ajouté une couche LSTM avec input dimension de 100 et 0.2 comme la valeur de dropout. En fin, la couche dense dont le nombre de neurones est égale au nombre de catégories qui est 10.

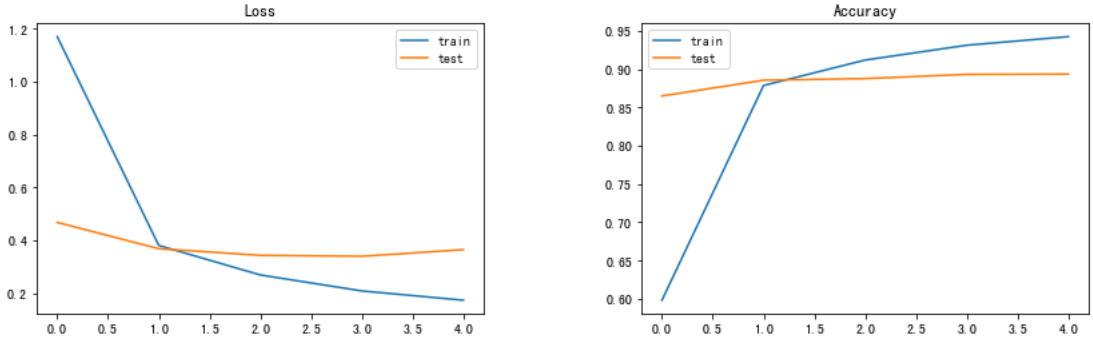
```
model = Sequential()
model.add(Embedding(self.MAX_NB_WORDS, self.EMBEDDING_DIM, input_length=X.shape[1]))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

(Code 1: définition de modèle LSTM)

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 250, 100)          5000000
_________________________________________________________________
spatial_dropout1d_1 (Spatial (None, 250, 100)          0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               80400
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1010
=================================================================
Total params: 5,081,410
Trainable params: 5,081,410
Non-trainable params: 0
_________________________________________________________________
None
```

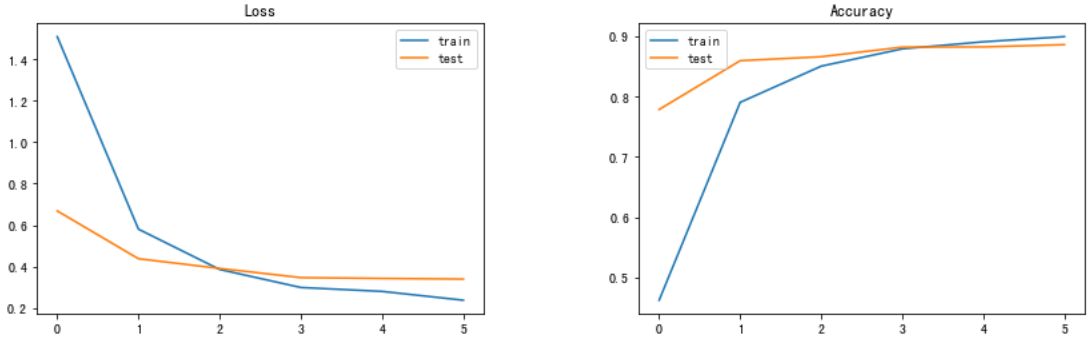
(Image 7: Résumé de modèle LSTM)

En plus, j'avais défini 5 epochs pour entraîner et 256 batch size puisque l'on a beaucoup de données. J'ai donc obtenu la précision de 0.88 et la f-mesure de 0.88, ainsi que les graphiques présentant la précision et la perte qui nous permet d'observer la performance d'un modèle. Plus la « loss » est faible, meilleur est le modèle. Cependant, ce modèle n'est pas assez performant vu que la tendance des valeurs de perte est à la hausse et les précisions n'augmentent plus après la deuxième époque.



(Image 8: Graphiques représentants la précision et la perte en fonction du nombre d'epochs avec epochs = 5, batch size = 256, et dropout = 0.2)

Il est possible qu'il est over-fitting en raison de trop de données. En conséquence, j'ai augmenté la pourcentage de « dropout » de 0.2 à 0.5 qui nous permet d'empêcher le over-fitting sur les données de training en abandonnant des unités dans un réseau de neurones. Et en même temps, j'ai modifié l'époque à 6 et changé l'output dimension de la couche à 64. Après ces changements, j'ai obtenu les résultats de la précision de 0.9 et la f-mesure de 0.89 avec les prétraitements de la suppression des ponctuations et des stopwords. Et les graphiques représentants la précision et la perte ont l'air mieux que les précédents.



(Image 9: Graphiques représentants la précision et la perte en fonction du nombre epochs avec epochs = 6, batch size = 256, et dropout = 0.5)

Avec ces paramètres, j'ai effectué avec les quatre prétraitements A (la suppression des ponctuations), B (la suppression des stopwords), C(la suppression des ponctuations et les

stopwords) et D(la suppression des stopwords et le filtrage des phrases de moins de 5 mots)et ci-dessous les résultats :

		A	B	C	D
4	LSTM	F1 : 0.88 Temps : 1530s	F1 : 0.88 Temps : 1546s	F1 : 0.88 Temps : 826s	F1 : 0.89 Temps : 931s

(Tableau 3: résultats sur différents combinaisons de la méthode LSTM)

En observant les résultats, j'observe que la précision(0.8) et la f-mesure(0.5) sont très bas sur la catégorie « 热水器(chauffe-eau) ». D'une part, ce corpus ne contient pas beaucoup de commentaires (575 au total) sur le produit chauffe-eau ; d'autre part, les commentaires pour la catégorie « chauffe-eau » n'est pas de bon qualité. Il existe beaucoup de commentaires telles que « 不错(pas mal) », «物流很快（la logistique est rapide) » qui sont des descriptions générales et pas assez concrètes pour cette catégorie. Cependant, la précision et la f-mesure pour la catégorie « 蒙牛(Mengniu lait) » sont souvent égales à 1. Cette situation pourrait peut-être s'expliquer par le fait que le marque « Mengniu » apparaît souvent dans les commentaires telles que « 蒙牛啊！我的最爱（J'aime bien Mengniu) », « 蒙牛还是挺好，实惠，最便宜（Mengniu est toujours très bon, abordable et moins cher) ».

La performance d'un modèle dépend de la qualité de données. Les résultats des catégories de « 平板(tablettes) » « 手机(téléphones portables) » ne sont pas élevés avec 0.8 et 0.86 comme la f-mesure respectivement en raison de l'ambiguïté des commentaires. Par exemple, le commentaire « 电池太卡 (La batterie ne fonctionne pas trop) » est catégorisé dans la catégorie « 平板(tablettes) » mais il peut aussi être un commentaire d'un téléphones portables.

Par conséquence, j'ai choisi les prétraitements de la suppression des stopwords et le filtrage des phrases de moins de 5 mots, puisqu'il est le plus performent parmi les quatre modèles LSTM.

Difficultés rencontrés

Dans l'ensemble du projet, le point qui était le plus difficile pour moi était de réussir à aller au delà des 87% de f-mesure. J'ai restées assez longtemps à 87% en testant les différentes combinaisons de méthodes. Enfin, J'ai constaté que c'était à cause des corpus test et train qui n'étaient pas équilibré et de bonne qualité. Le nombre de la catégories « chauffe-eux » est 575 couvrant seulement environ 1% du nombre total. Pendant ces essais, j'ai pu comprendre que l'équilibre des données et les choix de combinaisons de méthodes ont des impacts importants pour ce travail.

La bonne qualité de corpus est très importante pour l'entraînement d'un modèle de Machine Learning. En générale, j'ai pu faire des prétraitements tels que la suppression des singles inutiles et/ou des ponctuations, la suppression des stopwords et le filtrage des phrases pour obtenir un corpus de mieux qualité. Cependant, parfois, ce type de prétraitement n'est pas suffisant pour résoudre certains problèmes qui sont plus compliqués tels que la désambiguïsé et l'identification des expressions pragmatique ou contextuel dans ce corpus où il existe pleine d'expressions orales qui sont courtes mais ambigus et abstrait.

D'ailleurs, différents algorithmes ont diffèrent paramètres et/ou hyper- paramètres. Tuning les paramètres n'est pas une tâche facile et il n'y a pas de raccourcis. Lorsque l'on a obtenu un modèle pas assez performant, est-ce que l'on va modifier le nombre de l'époque, ou le batch size, ou bien d'ajouter des couches ? On peut bien sûr obtenir des graphiques pour

observer s'il est over-fitting ou under-fitting et on a encore des méthodes pour aider à identifier de bons paramètres. Cependant, cela est encore expérimental. C'est-à-dire que l'on a besoin de les tester et modifier petit à petit les (hyper-)paramètres avant de trouver un modèle performant et efficace. C'est exactement ce que j'ai fait pour trouver un bon modèle. Premièrement, j'ai identifié le problème selon les graphiques : Est-ce que les données sont trop ou pas suffisant pour entraîner ? est-ce qu'il est over-fitting ou under-fitting ? Ensuite, j'ai essayé de modifier des paramètres/hyper-paramètres pour résoudre ces problèmes et retester jusqu'à ce que j'aie un meilleur résultat.

Conclusion

L'objectif de ce projet est de créer un classifieur à partir d'un corpus des commentaires chinois de dix catégories et de comparer également les méthode de la Naïve Bayes et la Régression Logistique avec différentes représentations et la méthode LSTM avec différents prétraitements. En ce qui concernant la f-mesure, leurs résultats générés par ces trois algorithmes ne diffèrent pas beaucoup. La méthode de la Naïve Bayes peut obtenir un résultat très proche du ceux de la Régression Logistique mais avec beaucoup moins de temps d'entraînement et le modèle de LSTM a pris le plus de temps d'apprentissage.

Un corpus équilibre et de bonne qualité a des impacts très importants pour un système de classification, peu importe quel algorithme que nous choisissions. L'augmentation de nombre des données manquantes, la construction d'un corpus annoté ou d'un lexique pour ces catégories pourrait résoudre ce problème. Par exemple, dans le lexique de la catégorie de « téléphone-portables », on pourrait mettre les mots-clés tels que « 通话清晰，信号好（L'appel est clair et le signal est bon) » et dans celle de « tablettes », on met les mots-clés « 屏幕太小（Taille de l'écran est trop petite) » dans le lexique pour distinguer ces deux catégories.

L'un des avantages de la méthode LSTM est qu'il est utile pour les données textuelles qui ont des chaînes, puisque le sens d'un mot dépend de ceux qui l'ont précédé. Le modèle LSTM en général est légèrement plus performant de ceux de l'algorithme de la Naïve Bayes et la Régression Logistique avec le coût du temps long d'entraînement et de nombreux de données. Ce sont des inconvénients de l'algorithme LSTM.

Pour améliorer ce classifieur, on pourrait considérer à ajouter les données surtout les données de la catégorie « chauffe-eaux », ajouter un lexique et/ou annoter le corpus pour l'identification des mots spéciaux, ou bien utiliser des modèles pré-entraînés tels que le word2vec construit par des données dans ce domaine de produits de la maison.

Reference

Naive Bayes Document Scikit-learn https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html#sklearn.naive_bayes.MultinomialNB
Les Algorithmes de Naïves Bayes, Hausmane Issarane, 2019 https://le-datascientist.fr/les-algorithmes-de-naives-bayes
Logistic regression document Scikit-leanr
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
https://keras.io/api/layers/recurrent_layers/lstm/
Multi-Class Text Classification Model Comparison and Selection, Susan Li, 2018
https://towardsdatascience.com/multi-class-text-classification-model-comparison-and-selection-5eb066197568
LSTM layer Document Keras https://keras.io/api/layers/recurrent_layers/lstm/