PROJET TAL - RI

Charlotte Kruzic & Zoé Marquis

Étape 1

Méthodologie

Prétraitement des données

Le prétraitement des données est une étape cruciale dans tout projet d'apprentissage automatique. Dans notre travail, cette étape comprend les étapes suivantes :

1. **Nettoyage des données** : Suppression des caractères spéciaux, des ponctuations, etc., pour garantir des données propres et cohérentes.
2. **Tokenisation** : Division du texte en mots individuels, pour préparer les données à être traitées par les algorithmes d'apprentissage automatique.
3. **Suppression des mots vide**s : Élimination des mots courants et non informatifs qui n'apportent pas de valeur prédictive significative.
4. **N-grammes** : Exploration de la structure locale du texte en utilisant des n-grammes, qui sont des combinaisons de plusieurs mots consécutifs.
5. **Lemmatisation** : Réduction des mots à leur forme de base ou à leur "lemme", pour normaliser le texte et réduire la dimensionnalité de l'espace des fonctionnalités.
6. **Stemmatisation / Désuffixation** : Troncation des mots à leur radical ou à leur forme racine, pour regrouper les mots similaires et améliorer la généralisation du modèle.

Traits testés

Nous avons exploré plusieurs approches pour représenter nos données textuelles et les préparer à être utilisées par nos modèles d'apprentissage automatique. Ces approches comprennent :

1. **Sac de mots** (Bag of Words) : Une représentation simple mais efficace qui compte la fréquence des mots dans chaque document.
2. **TF-IDF** (Term Frequency-Inverse Document Frequency) : Une mesure de l'importance relative d'un terme dans un document par rapport à une collection de documents.

Pour ces 2 premières approches, les étapes 1 à 3 du prétraitements ont toujours été appliquées, les 3 dernières ont été testées à tour de rôle afin de tester leur nécessité dans l'amélioration du modèle.

3. **Informations statistiques** : Utilisation de caractéristiques statistiques supplémentaires extraites du texte, telles que la longueur du texte et le nombre de phrases.

Modèles évalués

Nous avons évalué la performance de différents modèles d'apprentissage automatique :

1. **Baseline** : Classificateur de base utilisant la stratégie de prédiction la plus fréquente.
2. **Naive Bayes multinomial**
3. **Arbre de décision** (CART)
4. **Régression logistique**
5. **K plus proches voisins** (KNN)
6. **Forêt d'arbres décisionnels** (Random Forest)
7. **Machine à vecteurs de support** (SVM)

Pour chaque modèle, nous avons réalisé une **validation croisée** à 5 plis pour évaluer sa performance de manière robuste et fiable.

Évaluation et visualisation

Nous avons utilisé différentes mesures d'évaluation, telles que l'***accuracy*/exactitude**, le ***F1-score***, la **précision** et le **rappel**, pour évaluer la performance de nos modèles sur plusieurs métriques.
Pour visualiser ces mesures, nous avons projeté les résultats sous forme de graphiques pour une comparaison facile entre les différents modèles et approches.

Travail réalisé

Le travail réalisé dans ce projet a débuté par la définition et la mise en place de toutes les possibilités décrites ci-dessus. Cela comprenait l'implémentation de **différents traits** avec **divers prétraitements**, la création de listes contenant tous les modèles à évaluer, ainsi que le **développement de fonctions** permettant de lancer l'**évaluation** (avec validation croisée) et la **visualisation** sur **plusieurs modèles pour plusieurs traits**, afin d'obtenir une vue d'ensemble lors de la comparaison en visualisant les différentes métriques.
Une fois cette phase préparatoire achevée, nous avons adopté une **approche incrémentale** pour tester les différentes combinaisons de modèles et de prétraitements. Étant donné que le nombre de possibilités de modèles à tester est immense, nous avons restreint notre exploration en nous concentrant sur des approches simples, en commençant avec seulement le sac de mots et/ou TF-IDF pour les deux colonnes possibles.
Nous avons aussi comparé les prétraitements pour évaluer leur impact sur les performances des modèles, comparé les performances de différents modèles de classification, et examiné l'effet de différents ensembles de traits sur les performances des modèles à chaque étape.
Cette approche méthodique nous a permis de dégager des tendances claires et de tirer des conclusions pertinentes sur les meilleures pratiques à suivre dans la préparation des données et le choix des modèles pour la classification de nos textes.

Pour une référence plus détaillée, nous vous invitons à consulter le notebook associé. Cependant, pour récapituler nos expérimentations, nous avons testé les combinaisons de différents traits 4 par 4. Voici un résumé :

- Les combinaisons conservées à chaque fois sont indiquées en vert.
- Ce que nous avons tenté d'ajouter à chaque fois est noté en gris.

Dans notre notebook, nous commentons progressivement les différentes performances, y compris l'exactitude, le score F1, la précision et le rappel, pour chaque combinaison testée. Nous espérons que cette synthèse vous sera utile.

n°	**headline**	**text**
1	TF-IDF, fréquence = 0.01	TF-IDF, fréquence = 0.01
2	Sac de mots, fréquence = 0.01	Sac de mots, fréquence = 0.01
3	TF-IDF, fréquence = 0.01	Sac de mots, fréquence = 0.01
4	Sac de mots, fréquence = 0.01	TF-IDF, fréquence = 0.01
Informations statistiques		
3	TF-IDF, fréquence = 0.01	Sac de mots, fréquence = 0.01
5	TF-IDF, fréquence = 0.01	Sac de mots, fréquence = 0.01 **Informations statistiques**

6	TF-IDF, fréquence = 0.01 **Informations statistiques**	Sac de mots, fréquence = 0.01
7	TF-IDF, fréquence = 0.01 **Informations statistiques**	Sac de mots, fréquence = 0.01 **Informations statistiques**
N-grammes		
5	TF-IDF, fréquence = 0.01	Sac de mots, fréquence = 0.01 Informations statistiques (N-gramme = 1)
8	TF-IDF, fréquence = 0.01	Sac de mots, fréquence = 0.01 Informations statistiques **N-gramme = 2**
9	TF-IDF, fréquence = 0.01	Sac de mots, fréquence = 0.01 Informations statistiques **N-gramme = 3**
10	TF-IDF, fréquence = 0.01	Sac de mots, fréquence = 0.01 Informations statistiques **N-gramme = 4**
Fréquence 0.1		
5	TF-IDF, fréquence = 0.01	Sac de mots, fréquence = 0.01 Informations statistiques
11	TF-IDF, **fréquence = 0.1**	Sac de mots, **fréquence = 0.1** Informations statistiques
12	TF-IDF, fréquence = 0.01	Sac de mots, **fréquence = 0.1** Informations statistiques
13	TF-IDF, **fréquence = 0.1**	Sac de mots, fréquence = 0.01 Informations statistiques
Fréquence 0.01 à 0.05		
5	TF-IDF, fréquence = 0.01	Sac de mots, fréquence = 0.01 Informations statistiques
13	TF-IDF, fréquence = 0.1	Sac de mots, fréquence = 0.01 Informations statistiques
14	TF-IDF, **fréquence = 0.05**	Sac de mots, **fréquence = 0.05** Informations statistiques
15	TF-IDF, fréquence = 0.1	Sac de mots, **fréquence = 0.05** Informations statistiques
Désuffixation		
5	TF-IDF, fréquence = 0.01	Sac de mots, fréquence = 0.01 Informations statistiques
16	TF-IDF, fréquence = 0.01 **Désuffixation**	Sac de mots, fréquence = 0.01 Informations statistiques
17	TF-IDF, fréquence = 0.01	Sac de mots, fréquence = 0.01 Informations statistiques **Désuffixation**
18	TF-IDF, fréquence = 0.01 **Désuffixation**	Sac de mots, fréquence = 0.01 Informations statistiques **Désuffixation**
Lemmatisation		

17	TF-IDF, fréquence = 0.01	Sac de mots, fréquence = 0.01 Informations statistiques Désuffixation
19	TF-IDF, fréquence = 0.01 **Lemmatisation**	Sac de mots, fréquence = 0.01 Informations statistiques Désuffixation
20	TF-IDF, fréquence = 0.01	Sac de mots, fréquence = 0.01 Informations statistiques Désuffixation **Lemmatisation**
21	TF-IDF, fréquence = 0.01 **Lemmatisation**	Sac de mots, fréquence = 0.01 Informations statistiques Désuffixation **Lemmatisation**

Par conséquent, nous avons choisi la chaîne 20 comme étant la plus appropriée, puisqu'elle maximise à la fois l'exactitude et le score F1 pour le modèle MultinomialNB. Bien que les scores de la chaîne 21 soient égaux à ceux de la chaîne 20, la chaîne 21 lemmatise également le titre, ce que ne fait pas la chaîne 20. Pour précision, par "chaîne", nous entendons la combinaison des différents traits ainsi que les prétraitements associés.
Les scores associés à la **chaîne 20** pour le modèle **MultinomialNB** sont les suivants :

- Exactitude : 0.8583050847457627
- F1-score : 0.8278586151680496
- Précision : 0.8238989637384575
- Rappel : 0.8351171338087339

Différentes approches

Nous avons également suivi une approche similaire à celle des travaux pratiques en essayant d'implémenter des modèles de **BiLSTM** et de **Transformers**.

Bi-LSTM

Pour l'architecture du modèle, nous avons utilisé un modèle Bi-LSTM pour analyser les titres et les textes dans les deux directions, ce qui permet d'améliorer la compréhension du contexte des mots. Nous avons également intégré une couche d'embedding en utilisant trois ensembles de vecteurs de mots pré-entraînés : model26 (similaire à celui utilisé dans le TP2), GloVe 42B et GloVe 6B. Le modèle comprend deux couches de BiLSTM et a été évalué en utilisant la méthode d'évaluation de l'exactitude avec une validation croisée à cinq plis.
Malheureusement, les résultats obtenus étaient moins satisfaisants que ceux du modèle 20 précédemment retenu. Voici les scores moyens pour tous les plis pour chaque modèle d'embedding pré-entraîné :

- Pour GloVe 42B :
    - Exactitude : 68.70 (+- 3.27)
- Pour GloVe 6B :
    - Exactitude : 66.12 (+- 4.35)
- Pour Model26 :
    - Exactitude : 67.21 (+- 4.00)

Transformers

Malheureusement, nous n'avons pas pu aller jusqu'au bout de notre expérimentation. Dans le notebook, vous trouverez le code associé à nos tentatives d'association avec Flaubert. Bien que toute l'implémentation jusqu'à l'entraînement soit fonctionnelle, nous avons rencontré un obstacle majeur lié

à la nécessité d'avoir un GPU pour exécuter les opérations requises. Les ordinateurs que nous avons utilisés ne disposent pas de suffisamment de puissance pour répondre aux exigences de traitement, ce qui a entraîné des erreurs lors de l'exécution du code. Même en essayant une solution distante comme Guacamole sur les ordinateurs de la faculté, le problème persiste. Il est devenu évident que l'utilisation d'un GPU est indispensable pour réaliser les tâches envisagées. Ces difficultés ont également été rencontrées lors du TP3, confirmant ainsi le besoin d'une solution adaptée pour l'exécution des modèles. Notons d'ailleurs que dans le TP3, la validation croisée n'avait pas été utilisée.
Malgré nos efforts pour préparer les données et le code nécessaire à l'entraînement des modèles, nous avons été limités par les capacités matérielles de nos ordinateurs. Pour illustrer notre progression, nous avons laissé le code correspondant dans le notebook, en veillant à ne pas l'exécuter (en le commentant) pour éviter de surcharger le Kernel.

Prédiction sur le jeu de test

Voilà les scores obtenus :

- Exactitude : 0.86
- F1-score : 0.86
- Précision : 0.86
- Rappel : 0.82

Les résultats liés à l'entraînement et ceux liés au test sont cohérents, il n’y a donc pas sur-apprentissage.

Analyse des résultats

Nous faisons face à un défi dans l'adaptation des données d'entrée et du modèle pour permettre à l'explainer Lime de prendre en charge plusieurs colonnes de texte simultanément. Contrairement au TP1 où nous avions un seul texte en entrée, notre modèle actuel nécessite plusieurs chaînes de texte pour effectuer des prédictions. Nous avons tenté de transformer les données ou le modèle pour résoudre ce problème, mais jusqu'à présent, nous n'avons pas réussi à trouver une solution efficace. Il semble que le problème réside dans la représentation des données, car Lime semble avoir des difficultés à traiter plusieurs colonnes de texte simultanément. Nous avons donc cherché des moyens de surmonter cette limitation afin de pouvoir utiliser l'explainer Lime de manière efficace avec notre modèle actuel.
Malheureusement, nos efforts n'ont pas donné les résultats escomptés. Toutefois, vous trouverez à la fin du notebook un résumé de ce que nous avons tenté, ainsi que les messages d'erreur rencontrés lors de nos essais.

Pistes d’amélioration

- Bi-LSTM avec GRU
- Transformers fonctionnels
- Tester d’autres préparations des données (il existe un grand nombre de combinaison, nous avons pu tester une petite partie d’entre elles)
- Combiner différents modèles

Étape 2

Pour cette étape du projet, nous avons commencé par télécharger et démarrer Solr.
$ ./bin/solr start

Indexation des documents

Nous avons ensuite créé un core nommé "Presse" afin d'y indexer les documents contenus dans le fichier 'train.tsv'.

Pour indexer ce fichier, il a fallu le convertir dans un format utilisable par Solr.
Nous avons choisi de le convertir en fichier JSON grâce aux commandes suivantes :
$ awk -F'\t' '{OFS=","; print}' train.tsv > train.csv
$ csvjson train.csv > train.json

Une fois le fichier créé, nous avons pu l'indexer dans notre core avec les commandes :
#Indexation
$ curl -X POST -H 'Content-Type: application/json' 'http://localhost:8983/solr/presse/update' --data-binary @train.json
#Validation de l'indexation
$ curl -X POST -H 'Content-Type: application/json' 'http://localhost:8983/solr/presse/update?commit=true'

Création de l'interface

Une fois l'indexation réalisée, nous avons pu retrouver l'ensemble des documents indexés sur la page HTML suivante : *http://localhost:8983/solr/presse/browse*

Modification de l'interface par défaut

Nous avons effectué quelques modifications afin d'obtenir une interface plus conviviale, de permettre les recherches, et d'offrir la possibilité de filtrer les articles par catégorie.

L'ensemble des modifications ont été réalisées dans les fichiers du dossier : *solr/server/solr/presse/conf*.

Dans le fichier *managed-schema*, nous avons changé les options linguistiques des quatre champs des documents ("category", "headline", "text" et "url"), en mettant leur type à la valeur 'text_fr', afin d'obtenir une recherche en français.

Dans le fichier *solrconfig.xml,* nous avons ajouté un champ de recherche par défaut afin de permettre les recherches sans spécifier de champ.
Nous avons également ajouté une facette avec le champ catégorie, permettant de filtrer les articles selon leur catégorie. Et nous avons ajouté une surbrillance sur le mot recherché dans les articles afin de mieux le repérer.

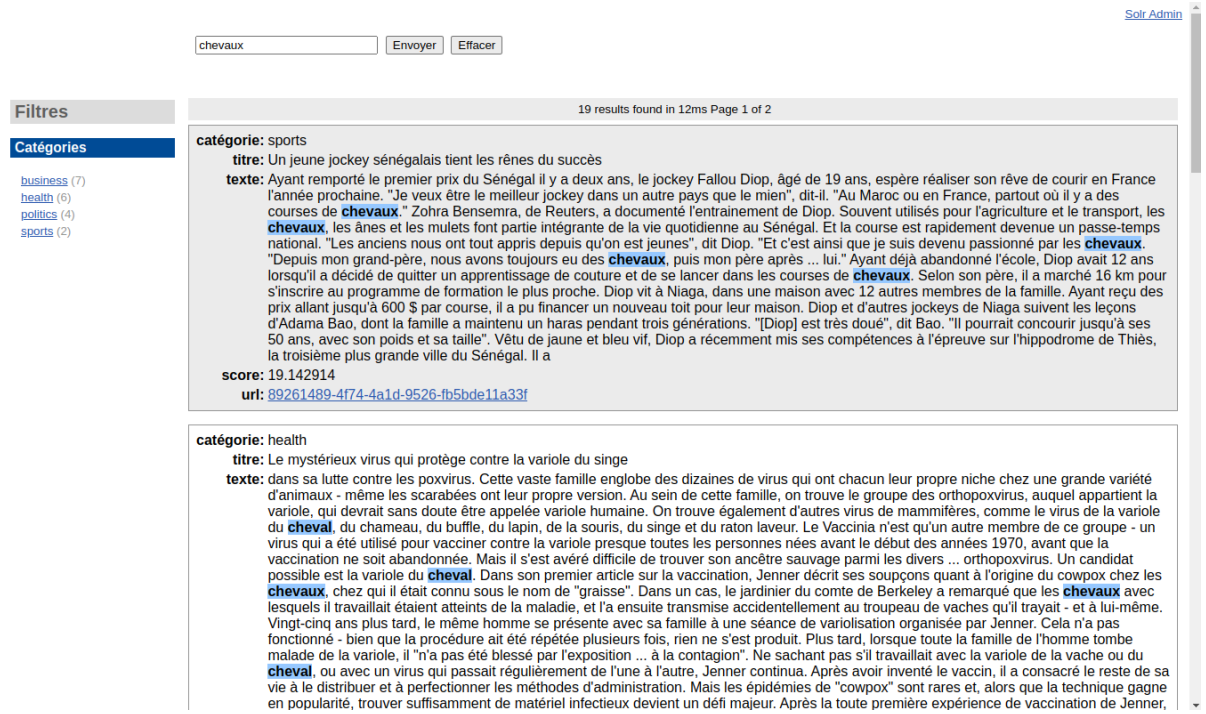
Enfin, nous avons ajouté le dossier *velocity* qui est une librairie donnant une base de templates. Nous avons apporté des modifications aux différents fichiers qu'il contient, notamment pour changer l'affichage de la barre de recherche, des articles, et de la facette. Nous avons pu changer les écritures, les fonds, mais aussi remplacer les étiquettes par défaut. Et nous avons ajouté un bouton pour réinitialiser la recherche.

Après avoir fait ces modifications et pour qu'elles soient prises en compte, nous avons effacé l'index et réindexé les articles.

#Effacer l'index
curl -H 'Content-type:application/json' -d '{"set-property": {"requestDispatcher.requestParsers.enableRemoteStreaming": true}, "set-property": {"requestDispatcher.requestParsers.enableStreamBody": true}}' http://localhost:8983/api/cores/presse/config
curl "http://localhost:8983/solr/presse/update?stream.body=%3Cdelete%3E%3Cquery%3E:*%3C/query%3E%3C/delete%3E"*
curl "http://localhost:8983/solr/presse/update?commit=true"

#Indexer les documents
curl -X POST -H 'Content-Type: application/json' 'http://localhost:8983/solr/presse/update' --data-binary @train.json
curl -X POST -H 'Content-Type: application/json' 'http://localhost:8983/solr/presse/update?commit=true'

Interface finale



Répartition

Charlotte Kruzic a pris en charge la réalisation de la partie 2 du projet, ainsi que la recherche de différents modèles de vecteurs pré-entraînés pour les Bi-LSTM. Zoé Marquis s'est quant à elle occupée de la préparation de la partie 1, comprenant la préparation des données, la mise en place des différentes fonctions et méthodes de visualisation, la recherche des différents traits et méthodes de prétraitement des données, ainsi que l'entraînement des modèles. Ensemble, nous avons toutes deux cherché à résoudre les problèmes liés aux transformers, liés à nos limitations matérielles.

Bibliographie

GloVe: Global Vectors for Word Representation
Bidirectional LSTM in NLP - GeeksforGeeks
Apache Velocity

NB : Les fichiers indexés se trouvent dans partie_2_RI/train.json.