Master 1 AIGLE et DECOL

Rapport de projet

HMIN232M - Méthodes Sciences des Données

Classification d'assertions selon leur valeurs de véracité (automatic fact-checking)

Auteurs
Sabri BENBRAHIM - 21604014
Bénédicte DAYNAC - 21605192
Yann DUFRESNE - 20055179
Llivia LANGEVIN - 21604582

Table des matières

Introduction

Dans le cadre de l'UE HMIN232M - Méthodes de la Science des Données, nous avons eu l'opportunité de réaliser ce premier projet de Sciences des Données.

Le but de ce projet est de définir un modèle de classification textuelle afin de prédire la vérité d'une assertion. Le corpus porté à notre disposition provenait du projet ClaimsKG. Sa taille atteignait presque 1 Go. Pour réaliser ce travail, il a été nécessaire de tester de manière empirique différents traitements et d'observer l'impact que cela avait sur les modèles de classifications.

Ce rapport regroupe donc nos réfléxions, nos choix. Ainsi dans un premier temps, nous expliquerons ce que nous avons réalisé pour préparer nos données, puis aux pré-traitements que nous leur avons appliqués. Par la suite, nous détaillerons l'impact des différentes features sur la prédiction. Enfin, nous nous intéresserons aux diverses classifications et à leurs valeurs.

Il est à noter que nous avons fait en sorte que notre Notebook soit le plus automatique, paramétrable, réutilisable et donc facile à utiliser. Cela nous permet de le faire tourner sans avoir à intervenir dessus.

1 Préparation des données

Le jeu de données initialement fourni est un CSV de plus de 730Mo encodé en UTF-8. La première étape du nettoyage a été importante. Afin de mieux visualiser ces données, nous avons choisi de la faire en partie "à la main" à l'aide d'un éditeur texte et d'Excel pour commencer.

Ainsi, nous avons tout d'abord supprimé les retours à la ligne à l'intérieur des colonnes (principalement dans les assertions). En effet, bien que le texte contenu dans une cellule de données soit délimité en étant encadré par des guillemets,

un retour à la ligne dans ce texte entraîne automatiquement une nouvelle ligne de données. Il résulte après ce premier traitement un fichier de plus de 39000 enregistrements (lignes).

Sur une ligne, un caractère de séparation délimite les changements de colonnes. La source utilisait la virgule, qui se retrouvait aussi très souvent dans le texte des assertions. Pour éviter les erreurs de décalage de contenu vers les colonnes suivantes, et après avoir supprimé les occurrences déjà existantes, nous avons choisi le caractère tabulation comme séparateur.

Concernant les colonnes présentes dans le fichier, deux ensembles d'informations se dégagent : un sur l'assertion (claim) et un autre sur la revue de l'assertion (review) et sa conclusion. Nous avons naturellement commencé par supprimer toutes les colonnes qui étaient vides ou en doublon, à savoir : claimReview_author, claimReview_author_name, claimReview_author_url.

Excel a permis de très rapidement supprimer les lignes strictement identiques (doublons). Concernant les valeurs présentes dans la colonne rating_alternateName, elles étaient très variées et ont été normalisées au moyen du script fournit sur ClaimsKG : -1 pour False, 1 pour True et 0 pour mixture. Après une reprise manuelle, l'ensemble des lignes n'ayant pu être qualifiées a été exclu afin d'alléger encore un peu le poids du jeu de données.

Une clé unique pour chaque enregistrement est générée pas l'union du nom du site de vérification (claimReview_source) avec l'id de la revue sur ce site (colonne id initialement sans nom).

Le travail suivant porta sur les colonnes extra_entities qui contenaient des informations relatives à d'autres colonnes (author, claim) sauvegardées sous la forme de JSON. Ces données ont été extraites et réparties sur 2 colonnes : entity et catégories qui viennent remplacer la colonne d'origine.

Si une quantité importante d'informations allant déjà dans le sens de la conclusion est prise en compte, un biais sera introduit dans le modèle. Pour cette raison le texte de revue de l'assertion (body) ainsi que son titre (title) et la colonne extra_entities associée ont été exclus.

Enfin, un dernier groupe de colonnes a été supprimé (url, dates) : creativeWork_author_sameAs, claimReview_datePublished, creativeWork_datePublished, extra_refered_links, rating_bestRating, rating_ratingValue, rating_worstRating.

Nous avons divisé le jeu de données en deux. Le premier pour True vs False, le second pour True-False vs Mixture. Pour commencer nos recherches, étant donné que ces deux jeux de données étaient encore de taille trop importante, nous avons choisi le jeu True vs False que nous avons divisé en trois, en fonction des reviewers : 9 sources au total dont 2 gros reviewers (politifact et snopes). Le plus petit des 3 jeux permis l'exécution de Notebooks sur nos machines respectives.

Résumé des colonnes conservées pour l'analyse par le Notebook :

- **claimReview_source_id** : clé unique de l'enregistrement
- **claimReview_claimReviewed** : correspond à l'assertion
- **claimReview_url** : lien vers la page de la revue sur le site de vérification
- **creativeWork_author_name** : le nom de l'auteur de l'assertion
- **extra_author_categories** : extraction des catégories relatives à l'auteur
- **extra_claimReview_claimReviewed_entity** : extraction des entités détectées dans l'assertion
- **extra_claimReview_claimReviewed_categories** : extraction des catégories relatives aux entités de l'assertion
- **extra_keywords_entity** : extraction des entités mots clés
- **extra_keywords_categories** : extraction des catégorie des mots clés
- **extra_tags** : d'autres mots clés
- **rating_alternatename** : conclusion du fact-checking du site vérification
- **true_false_mixture** : résultat de la normalisation des valeurs de la colonne précédente

2 Pré-traitements

2.1 Nettoyage des claims

Tout d'abord, pour nettoyer les assertions, nous utilisons la méthode *preclean_data* qui procède en plusieurs étapes.

En premier, l'**encodage des caractères** est normalisé pour éviter d'éventuels problèmes d'encodage. En second, les **balises HTML** issues de l'extraction depuis le site de revue sont enlevées car cela n'a pas de sens de les garder dans notre cas, seul le texte est pertinent. Enfin, les **contractions anglaises** sont remplacées par leur forme complète. Nous diminuons ainsi le nombre d'occurrences et évitons les termes comme "isn".

2.2 Nettoyage des tokens

Nous appliquons la méthode token_data qui consiste à transformer un ensemble de chaînes de caractères en un ensemble de tableaux de **tokens**.

Ces tokens sont à leur tour nettoyés : tout est mis en **minuscules** (cela nous

permet de regrouper des mêmes mots mais de casses différentes), les **nombres** sont écrits en toutes lettres et les **symboles** qui ne sont pas des lettres sont supprimés (par exemple la ponctuation non pertinente).

Le but de toutes ces opérations est principalement d'enlever les informations non pertinentes et de regrouper les informations identiques. Cela nous permet de regrouper par exemple "Nom" avec "nom" et "deux" avec "2".

2.3 Préparation au classifieur

Les tokens sont "mergés", assemblés à nouveau en chaînes de caractères, puis **vectorisés**. Le texte est transformé en dictionnaire de mots avec leur fréquence. Nous utilisons le vectorizer TFIDF qui pondère les mots afin de réduire l'impact des mots trop fréquents et donc moins signifiants. De plus, nous excluons du dictionnaire les mots présents une seule fois en utilisant le paramètre *df_min*. Cette action nous a permis de réduire la taille du vecteur tout en améliorant sa pertinence.

Le classifieur va s'entraîner a obtenir la conclusion a partir du texte. Pour cela nous mettons donc en place les jeux d'apprentissage.

3 Impact des features sur la prédiction

Cette partie retrace notre processus de sélection des features. A cette fin, nous utilisons la méthode nommée *do_classifier* qui nous permet d'exécuter un modèle de classification sur un dataframe donné et d'afficher toutes les informations relatives à son exécution : accuracy, F1-score, matrice de confusion, courbe ROC. Par défaut, c'est le LinearSVC car il est rapide et donne de bons résultats. L'exécution de cette méthode nous sert donc d'indicateur qui nous permet de mesurer l'impact de l'application de features.

Pour évaluer les modèles l'**accuracy** est un élément important, mais elle présente certaines limites, notamment lorsque les données ne sont pas équilibrées.

Le **F1-Score** est un indicateur tenant compte de la précision et du rappel. Nous l'utilisons car il est conseillé quand il y a une distribution déséquilibrée des classes et que celles-ci sont binaires. Ce score s'appliquant à chaque classe, pour n'avoir

qu'une seule valeur à comparer d'un modèle à l'autre, nous utilisons le F1-Score macro-moyen.

La **matrice de confusion** permet une évaluation détaillée en affichant l'ensemble des résultats pour chaque classe. Elle affiche le nombre de faux négatifs, faux positifs, vrais négatifs et vrais positifs.

Enfin, la **courbe ROC** ("Receiver Operating Characteristics") permet de visualiser graphiquement les performances d'un classifieur. En effet, elle affiche le taux de vrais positifs en fonction du taux de faux positifs.

A chaque fois que nous évaluons un traitement, l'ensemble des résultats est comparé au meilleur score [1] du traitement précédent. L'option ayant obtenu le meilleur score est retenue. Ce nouveau score et le jeu de données résultant du traitement sont réutilisés pour exécuter le traitement suivant.

A chaque traitement est associé une variable, qui mémorise l'option qui a été retenue. Ces variables sont utilisées dans certaines fonctions, ce qui nous permet d'automatiser le processus de décision final, d'afficher l'ensemble des options retenues au cours de l'évaluation des features et éventuellement de reproduire le résultat ultérieurement.

Initialement, le jeu de données True vs False complet donne un F1-Score de 48.497%.

3.1 Suppression des stop words

Le premier traitement que nous testons est celui de la suppression des stop words, les "mots vides", trop communs pour que le fait de les indexer ajoute du sens.

Étonnamment, le F1-Score a légèrement diminuée et passe à 46.514%. La suppression des stop words n'est donc pas retenue.

3.2 Lemmatisation

De la même manière, nous testons l'effet de la lemmatisation, qui permet de raccourcir les mots en restituant leur forme canonique.

De nouveau, le F1-Score diminue et passe à 46.790%. Nous ne le prenons donc pas en compte.

1. Lorsque les données ne sont pas équilibrées, le F1-Score est utilisé. Après avoir appliqué les méthodes de resampling, c'est l'accuracy qui est retenue. Ainsi nous utilisons le terme *score* pour être plus générique.

3.3 Ajout des méta-données

Nous souhaitons également évaluer si l'ajout des méta-données extraites, c'est à dire les colonnes liées à l'auteur ou les colonnes "extra", est bénéfique.

En ajoutant uniquement les auteurs, le score augmente et atteint 53.367%. En ajoutant toutes les méta-données, le F1-Score diminue avec 51.126%. Ainsi, seules les colonnes "extra" liées à l'auteur sont ajoutées.

3.4 Méthodes de resampling

Nous avons pu constater que les conclusions des fact-checking aboutissaient beaucoup plus souvent à False qu'à True. Ainsi il est nécessaire d'équilibrer notre jeu de données.

Pour cela, nous avons fait appel à 2 méthodes différentes de resample : une proposée par dataframe et l'autre par scikitlearn. Puis nous avons évalué leur impact. Pour rappel, le processus d'upsampling permet de dupliquer de manière aléatoire des lignes de la "classe minoritaire" pour renforcer son signal. Le downsampling est le même processus mais en enlevant des lignes de la classe majoritaire.

	Accuracy	**F1 Score**
Avant resample	75.329%	53.367%
Downsampling	64.748%	64.686%
Upsampling	75.325%	75.318%
Downresample	65.657%	65.650%
Upresample	75.325%	75.318%

On constate donc que l'upsampling et l'upresample sont les méthodes qui améliorent le plus le score. On peut remarquer que le downsampling et le downresampling sont les seuls cas où l'accuracy diminue alors que le F1-Score augmente. Pourtant, il est essentiel d'équilibrer les données. Ainsi on peut noter l'importance d'avoir utilisé le F1-Score.

3.5 Autre

Concernant le POS-Tagging, nous avons choisi de ne pas l'appliquer car nous avons considéré que cela serait une perte d'information. Néanmoins, si nous avions eu plus de temps nous aurions pu évaluer les natures de mots les moins pertinentes pour la prédiction afin de les éliminer.

Étant donné la taille du vecteur de TFIDF il était difficile d'y appliquer les N-Grams. Cependant, si nous avions pu la réduire considérablement, cela aurait pu être envisageable.

3.6 Résumé des résultats

Nous avons donc appliqué le même processus au jeu de données True-False vs Mixture. Le tableau ci-dessous regroupe les valeurs de F1-score que nous avons obtenues.

	T vs F	**T-F vs M**	
Initial	48.497%	59.670%	
Suppression des Stop Words	46.514%	59.073%	
Lemmatisation	46.790%	58.753%	
Ajout des méta-données liées à l'auteur	**53.367**%	**64.100**%	Retenu
Ajout des autres méta-données	51.126%	63.712%	
Downsampling	64.748%	68.895%	
Upsampling	75.325%	73.025%	
Downresample	65.657%	70.064%	
Upresample	**75.318**%	**73.025**%	Retenu

Les traitements retenus pour améliorer le score sont les mêmes pour les deux jeux de données.

4 Recherche de modèles de classification

4.1 Modèles de classification

Grid Search est une méthode de recherche d'hyper-paramètres pour optimiser l'évaluation pour un modèle donné et un jeu de donnée. Nous avons utilisé un GridSearchCV afin de déterminer les meilleurs hyper-paramètres pour chacun des classifeurs suivants :

— GaussianNB
— MultinomialNB
— LinearSVC
— LogisticRegression
— DecisionTreeClassifier
— RandomForestClassifier
— SGDClassifier

Nous avons sélectionné ces classifieurs car nous souhaitions d'une part appliquer ceux que nous avons étudié en cours mais aussi en découvrir d'autres. D'ailleurs, nous avons trouvé ces derniers sur Internet. La recherche est un processus long, les résultats sont sauvegardés dans un fichier afin d'être archivés.

Ensuite, nous appliquons ces paramètres à chaque modèle de classification afin de rechercher celui qui donnera la meilleure accuracy. Cette étape utilise pour chacun de ces modèles une méthode de Cross-Validation. Il s'agit d'une méthode d'estimation en coupant la donnée en plusieurs parties en liaison par une sous méthode.
Il existe deux techniques de validation : le Train Split Test et le K-Fold.
Avec le Train Split Test, la donnée est coupée en 70% de jeu d'entraînement et 30% de test.
Avec le K-Fold, la donnée est coupée en k groupes et effectue k fois le test avec, dans ce groupe de k, un qui deviendra le jeu de test et le reste sera un jeu d'entraînement.
Ici, nous utilisons K-Fold avec le nombre de groupes à fixé à 5.

Modèle de classification	**True vs False**	**True-False vs Mixture**
GaussianNB	77.488627%	72.252368%
MultinomialNB	77.395073%	73.395706%
LinearSVC	83.101483%	77.257629%
LogisticRegression	84.013906%	77.407122%
DecisionTreeClassifier	63.316127%	NA
RandomForestClassifier	63.787304%	NA
SGDClassifier	78.988118%	NA

Le LogisticRegression et LinearSVC se sont révélés être les plus performants sur nos 2 jeux de données atteignant une précision entre 84% et 83% pour le jeu True vs False et 77% (pour les deux) pour True-False vs Mixture. Nous avons tout de même pu observer une différence importante de précision entre les divers classifieurs. Par exemple, il y a environ 20 points d'écart entre le LogisticRegression et le DecisionTreeClassifier. Ainsi, nous retenons le LogisticRegression qui devance très légèrement LinearSVC.

4.2 Sélection de variables

Il existe plusieurs méthodes de sélection de variables. Dans notre cas, nous souhaitions utiliser une méthode de sélection de variables univariée qui utilise SelectKBest. Malheureusement, après plusieurs essais, nous ne sommes pas arrivés

à l'éxécuter à cause de certaines erreurs.

5 Bilan

Après avoir rencontré plusieurs problèmes liés à une mémoire (RAM) insuffisante sur nos machines respectives, nous sommes passés sur Colaboratory, un service proposé par Google permettant de créer et partager des notebooks en ligne. Ce service est gratuit et nous permet d'avoir jusqu'à 25 GB de RAM et 107 GB de disque. Malgré les temps d'exécution très longs (plusieurs heures) cet outil nous a permis d'exécuter des notebooks entiers qui ne pouvaient pas fonctionner sur nos ordinateurs personnels.

Nous avons réussi à atteindre un score de 84% avec le classifieur LogisticRegression. L'apprentissage automatique dans le cadre du Fact-Checking ne met pas nettement en évidence l'impact des features. L'auteur de l'assertion peut légèrement influer sur la prédiction. L'équilibrage du jeu de données améliore le plus.

Comme tout projet, celui-ci est améliorable. Nous pourrions par exemple réaliser une sélection des variables pour rehausser nos résultats. L'utilisation des méthodes de N-Grams et de Pos-Tagging pourrait également être envisageable.