RAPPORT DE PROJET : CLASSIFICATION DE TWEETS

Mia Swery
Eva Radu

I. INTRODUCTION

Notre première idée pour le projet était de créer un classificateur qui indique s'il y a du cyber-harcèlement au sein de conversations sur des réseaux sociaux. Cependant, nous avons dû faire face à un problème de taille à savoir : un total manque de ressources sur le sujet. Cela nous a amené à notre seconde idée : détecter si des tweets peuvent être considérés comme étant sexistes ou non.

Étant donné la complexité de notre jeu de données, nous avons préféré nous cantonner à de la simple classification et non pas distinguer les différents types de sexismes (religion, travail…).

Afin de mener à bien notre tâche, nous avons utilisé différents types de classification et avons essayé de déterminer la meilleure.

II. ORGANISATION DU TRAVAIL

Pour ne pas perdre le fil, nous avons préféré créer un repository GitHub dans lequel nous avons posté nos fichiers .ipynb. Toutefois, Mia a régulièrement eu des problèmes avec l'environnement jupyter. Nous avons donc travaillé ensemble tout au long du projet et Eva postait nos avancées sur GitHub.

III. LES DONNÉES

Nous avons décidé de partir du jeu de donné présent sur le répertoire github suivant: https://github.com/patriChiril/An-Annotated-Corpus-for-Sexism-Detection-in-French-Tweets

En particulier, nous avons commencé par exploiter les données présentes dans le fichier "*corpus_SexistContent.csv*". À l'origine, nous y trouvions uniquement les identifiants des tweets et les catégories associées (0 = non sexiste, 1 = sexiste). Nous savions dès lors que notre projet s'appuierait sur de l'apprentissage supervisé.

C'était donc à nous d'extraire ces tweets depuis l'API Twitter. Nous effectuons cette opération dans notre fichier *Récupération_Tweets.ipynb*. Malheureusement, sur les 12274 tweets annotés, seuls 3150 sont encore disponibles. Nous avons de ce fait dû utiliser un jeu de données peu équilibré puisque sur ces 3150 tweets, 2161 sont répertoriés comme étant non-sexistes et 989 comme étant sexistes.

Une fois ces tweets récupérés, il nous a fallu les "nettoyer" dans le but d'obtenir des résultats plus fiables. En effet, nous avons décidé de retirer les caractères de ponctuations, des mots vides (ou stop words), les urls et la majorité des émojis français.

- Ponctuations : !()-[]{};:'"\,<>./?#$%^&*_~'@“”‘’«»

- Mots vides : une liste de verbes, de déterminants et de pronoms qui ne nous apprennent rien sur le caractère sexiste d'un tweet et ne valent donc pas la peine d'être pris en compte

Nous avons également procédé à une lemmatisation de nos phrases, en s'appuyant sur la version française du stemmer Snowball. Il n'est pas parfait mais il nous aide tout de même à remonter jusqu'à la racine de certains mots !
Nous avons sauvegardé ces tweets nettoyés dans notre fichier *my_csv_clean.csv*
Voici un tableau récapitulatif de notre jeu de données :

	Non-Sexiste	Sexiste
Nombre de tweets	2161	989
Exemples non nettoyés	Les jumelles Estelle et Delphine Cascarino convoquées par Corinne Diacre pour le match amical contre l'Italie au Vélodrome le 20/01. #Bleues #France2019 https://t.co/37Tlmhhlli	"Répète après moi : Les Hommes Ne Souffrent Pas De Sexisme Éduque-toi merci https://t.co/PHGubndWqy"
Exemples nettoyés	jumel estel delphin cascarino convoque corinn diacr match amical contr ital velodrom 2001 bleu france2019	repet apre homm ne souffrent pas sexism eduquetoi merc

De plus, tout au long du projet nous avons séparé notre jeu de données dans les proportions suivantes : on a 30% de test et 70% de tweets dédiés à l'apprentissage. Ces tweets sont sélectionnés au hasard à chaque fois que l'on divise notre jeu de données.

IV. CONCEPTION ET IMPLÉMENTATION

Dans le but d'obtenir les meilleures prédictions possibles, nous avons testé plusieurs modèles :

a) Support Vector Classification

La SVM est une technique d'apprentissage supervisé (on connaît déjà les valeurs de sortie) souvent employée pour la classification. Pour rester simple, on essaie de séparer les données appartenant à différentes catégories à l'aide de Support Vectors qui aideront à tracer une 'ligne de décision' maximisant la distance entre les points de différentes catégories. Le but est donc de construire le meilleur hyperplan. Ce dernier peut être créé suivant différentes fonctions dites "noyau" (ou "kernel"). On a donc testé le modèle SVM avec différents paramètres pour chercher quel kernel convenait le mieux à nos données de départ. En effet, chaque type de kernel possède sa propre fonction noyau :

i) ***Radial Basis Function (RBF) Kernel*** : $k(x,y) = exp\left(-\frac{||x-y||^2}{2\sigma^2}\right)$

ii) ***Linear Kernel :*** $k(x,y) = x^T y + c$

iii) ***Polynomial Kernel :*** $k(x,x') = (\alpha x^T . x' + \lambda)^d$

iv) ***Sigmoid Kernel :*** $k(x,y) = tanh(\alpha x^T y + c)$

Premièrement, il nous était nécessaire d'utiliser la fonction CountVectorizer(), qui nous a permis de transformer les tweets du jeu de donnés d'entrainement sous forme de tokens et de construire un vocabulaire des mots connus.

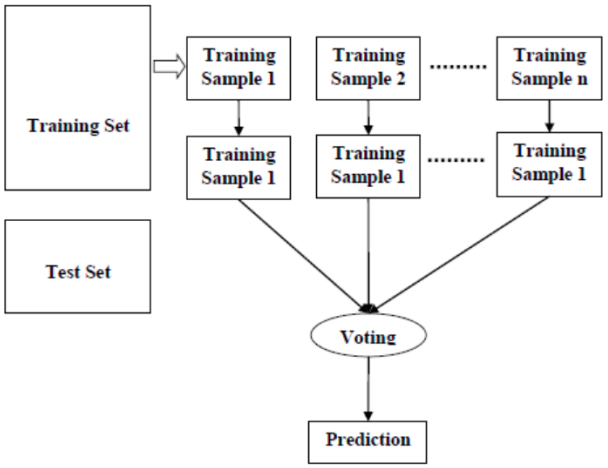
À la suite de cela, nous avons utilisé TfidfTransformer() pour que les tokens qui sont les plus fréquents soient aussi ceux qui sont les plus importants lors de notre classification.

Ce sont ces deux fonctions que nous avons utilisées dans tous nos modèles.

b) Random Forest Classification

Dans cette méthode, on a recours à une succession d'arbres construits différemment (ie les données entraînées et testées diffèrent à chaque itération) pour prédire si un tweet sera sexiste ou non.

Schéma explicatif :



(source : https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_random_forest.htm)

c) Naive Bayes

Nous avons également décidé d'utiliser une classification Naive Bayes. Cette classification est une approche qui se base sur les probabilités, et en particulier sur le théorème de Bayes. En effet, grâce à ce théorème, et en s'appuyant sur une hypothèse d'indépendance des classes, on peut calculer des probabilités conditionnelles qui nous permettront de déterminer à quelle classe appartient un tweet.

Pour utiliser ce modèle, nous avons utilisé les deux approches suivantes :

i) Le classifier MultinomialNB() proposé en python

Comme précédemment, après avoir tokenisé nos tweets et après avoir transformé nos données avec tfidf, nous avons pu tester notre modèle en utilisant MultinomialNB().

De plus, dans le but d'améliorer nos résultats, nous avons décidé de répéter cette procédure en équilibrant de plusieurs manières nos données. En effet, notre jeu de données initiales contient beaucoup plus de tweets non-sexistes que de tweets sexistes. Cela peut "influencer" nos prédictions dans le sens où notre modèle apprend davantage depuis les tweets non sexistes et donc aura tendance à prédire plus fréquemment qu'un tweet est non sexiste.

Nous avons donc testé notre modèle en utilisant la méthode de OverSampling pour forcer les tweets sexistes à être de même nombre que les tweets non sexistes. De la même façon, nous avons utilisé de l'UnderSampling pour forcer les tweets non sexistes à être de même nombre que les tweets sexistes. Enfin, nous avons mêlé ces deux méthodes.

ii) Calcul des probabilités manuellement

Afin de comprendre davantage comment marchait ce modèle, et dans le but d'essayer d'améliorer nos résultats, nous avons décidé d'implémenter manuellement la méthode Naïve Bayes. Pour cela, nous avons dû calculer les probabilités $P(Sexist)$ et $P(NotSexist)$ qui correspondent respectivement à :

$$P(Sexist) = \frac{Nombre\ de\ tweets\ sexistes\ dans\ le\ jeu\ de\ données\ d'entrainement}{Nombres\ total\ de\ tweet\ dans\ le\ jeu\ de\ donnés\ d'entrainement}$$

$$P(NotSexist) = \frac{Nombre\ de\ tweets\ non\ sexistes\ dans\ le\ jeu\ de\ données\ d'entrainement}{Nombres\ total\ de\ tweet\ dans\ le\ jeu\ de\ donnés\ d'entrainement}$$

De plus, pour chaque tweet présent dans le jeu de données d'entrainement, nous avons calculé des probabilités pour chacun des mots contenus dans le tweet, et ce, pour chaque catégorie.

Puis, lorsque l'on a voulu tester notre modèle avec des tweets présents dans le test set, nous avons multiplié les probabilités des mots et avons comparé les résultats obtenus pour chaque catégorie.

Exemple : supposons que l'on ait un tweet de 3 mots : "word1 word2 word3". Alors, nous avons comparés les deux résultats suivants :

$$P(tweet \mid Sexist) = P(word1 \mid Sexist) * P(word2 \mid Sexist) * P(word3 \mid Sexist) * P(Sexist)$$
$$P(tweet \mid NotSexist) = P(word1 \mid NotSexist) * P(word2 \mid NotSexist) * P(word3 \mid NotSexist) * P(NotSexist)$$

Sachant que, par exemple :

$$P(word1 \mid Sexist) = \frac{Nombre\ de\ tweets\ dans\ le\ jeu\ de\ données\ d'entrainement\ qui\ contiennent\ le\ mot\ word1}{Nombre\ de\ tweets\ sexistes\ dans\ le\ jeu\ d'entrainement}$$

Si la probabilité $P(tweet \mid Sexist)$ est plus grande que $P(tweet \mid NotSexist)$, alors notre tweet est classifié comme étant sexiste, et dans le cas inverse notre tweet est classifié comme étant non-sexiste.

d) Vecteurs de mots : Word2Vec

Le “word embedding” ou vectorisation de mots est une méthode d'apprentissage qui permet de transformer chaque mot en un vecteur de nombres réels. Ces nombres réels sont calculés en fonction du contexte et de la similitude d’un mot avec d’autres, d’où la notion de “vecteur de mot”.

Une implémentation assez populaire du word embedding est Word2Vec. Cette technique s’appuie des réseaux de neurones artificiels à deux couches entraînés pour reconstruire le contexte linguistique des mots à partir de vecteurs de mots. Elle peut être implémentée de deux manières différentes :

- *SG* :

Le modèle skip-gram apprend en considérant un mot tiré d'une phrase. Pour entraîner le réseau de neurones, on extrait un mot d’une phrase et c’est ce mot qui constituera l’entrée du réseau. Le reste de la phrase sera notre sortie.

- *CBOW* :

Le modèle Continuous Bag-of-word fait le contraire du modèle SG. Il apprend à reconnaître un mot à partir d'un contexte.

Nous aurions voulu combiner ces méthodes à des réseaux de neurones, malheureusement cela n’a jamais marché (malgré de nombreuses tentatives de notre part !). Nous avons donc décidé d’exploiter ces modèles autrement. En effet, pour ces deux modèles, nous avons décidé d’utiliser la méthode suivante : pour chaque tweet présent dans le jeu de données de test, nous avons trouvé les mots les plus proches et avons regardé si ces mots étaient plutôt présents dans des tweets sexistes ou des tweets non sexistes. Et c’est ainsi que nous avons prédit les classes pour chaque tweet.

e) Réseau de neurones keras

Enfin, même si nous n’avons pas réussi à utiliser des réseaux de neurones avec Word2Vec, nous n’avons pas abandonné l’idée de construire un réseau de neurones visant à classifier nos tweets pour autant ! Nous avons réussi à le faire en ayant recours à la libraire Keras de Tensorflow.
Nos paramètres initiaux sont :

- Taille du vocabulaire : 1000
- La dimension d’embedding : 16
- La plus grande taille de phrase : 120
- Si les phrases sont trop courtes ont les coupe en fin de phrase (paramètre “post”)
- Si les phrases sont trop courtes ont les “remplies” en fin de phrase (paramètre “post”)

Comme précédemment, nous avons testé cette méthode en ayant également recours à l’oversampling et à l’undersampling. De plus, toujours dans le but de comparer nos modèles et de déterminer le meilleur, nous avons décidé de tester le modèle avec deux tailles de vocabulaire différentes (1000 et 4000), avec deux dimensions d’embedding (16 et 18) et deux tailles de plus grande phrase (120 et 80).

V. RESULTATS OBTENUS

a) ***Accuracy et Balance Accuracy***

Modèle	*Accuracy*	*Balanced Accuracy*
Support vector avec paramètre kernel = 'rbf'	0.753968253968254	0.6461803867632587
Support vector avec paramètre kernel = 'sigmoid'	0.7507936507936508	0.6655896894748699
Support vector avec paramètre kernel = ‘poly’	0.7380952380952381	0.6043620540425179
Support vector avec paramètre kernel = ‘linear’	0.7206349206349206	0.6568691383787244
Random Forest Classifier	0.7523809523809524	0.6435897131881292
Word2Vec - CBOW	0.7005291005291006	-
Word2Vec - Skip Gram	0.68994708994709	-
Naive Bayes avec MultinomianlNB()	0.7174603174603175	0.5409660097898557
Naive Bayes avec MultinomianlNB() + OverSampling	0.7469512195121951	-
Naive Bayes avec MultinomianlNB() + UnderSampling	0.7422145328719724	-
Naive Bayes avec MultinomianlNB() + OverSamping et UnderSampling	0.7591463414634146	-
Naive Bayes - Approche manuelle	0.6402116402116402	0.7217465753424658
Naive Bayes - Approche manuelle + OverSampling	0.7697674418604651	-
Naive Bayes - Approche manuelle + UnderSampling	0.7298657718120806	-
Réseau de neurones Tensorflow keras	0.7428571428571429	-
Réseau de neurones Tensorflow keras + OverSampling	0.48604651162790696	-
Réseau de neurones Tensorflow keras + UnderSampling	0.5105633802816901	-
Réseau de neurones Tensorflow keras avec vocab size = 4000	0.7714285714285715	-
Réseau de neurones Tensorflow keras avec embedding_dim = 18	0.7619047619047619	-
Réseau de neurones Tensorflow keras avec max_length = 80	0.7597883597883598	-

b) Matrices de confusion et métriques

Étant donné que nous avons beaucoup de modèles différents, nous avons décidé de ne répertorier ici que les résultats pour les modèles ayant les meilleures accuracy. Tous les autres résultats sont présents dans nos différents fichiers jupyter.

1) Model SVC avec paramètre kernel = 'rbf' :

- *Matrice de confusion :*

Predicted Labels True Labels	Not Sexist	Sexist
Not Sexist	404	33
Sexist	122	71

- *Métriques :*

	precision	recall	f1-score	support
Not Sexist	0.77	0.92	0.84	437
Sexist	0.68	0.37	0.48	193
Accuracy			0.75	630
Macro avg	0.73	0.65	0.66	630
Weighted avg	0.74	0.75	0.73	630

2) Random Forest Classification:

- *Matrice de confusion :*

Predicted Labels True Labels	Not Sexist	Sexist
Not Sexist	404	33
Sexist	123	70

- *Métriques :*

	precision	recall	f1-score	support
Not Sexist	0.77	0.92	0.84	437
Sexist	0.68	0.36	0.47	193
Accuracy			0.75	630
Macro avg	0.72	0.64	0.66	630
Weighted avg	0.74	0.75	0.73	630

3) Word2Vec - CBOW :

- *Matrice de confusion :*

Predicted Labels True Labels	Not Sexist	Sexist
Not Sexist	549	118
Sexist	165	113

- *Métriques :*

	precision	recall	f1-score	support
Not Sexist	0.77	0.82	0.80	667
Sexist	0.49	0.41	0.44	278
Accuracy			0.70	945
Macro avg	0.63	0.61	0.62	945
Weighted avg	0.69	0.70	0.69	945

4) Naive Bayes avec MultinomialNB() et OverSampling et UnderSampling :

- *Matrice de confusion :*

Predicted Labels True Labels	Not Sexist	Sexist
Not Sexist	520	136
Sexist	180	476

- *Métriques :*

	precision	recall	f1-score	support
Not Sexist	0.74	0.79	0.77	656
Sexist	0.78	0.73	0.75	656
Accuracy			0.76	1312
Macro avg	0.76	0.76	0.76	1312
Weighted avg	0.76	0.76	0.76	1312

5) Naive Bayes avec l'Approche manuelle et l'OverSampling :

- *Matrice de confusion :*

Predicted Labels True Labels	Not Sexist	Sexist
Not Sexist	445	200
Sexist	97	548

- *Métriques :*

	precision	recall	f1-score	support
Not Sexist	0.82	0.69	0.75	645
Sexist	0.73	0.85	0.79	645
Accuracy			0.77	1290
Macro avg	0.78	0.77	0.77	1290
Weighted avg	0.78	0.77	0.77	1290

6) Réseau de neurones Tensorflow keras avec vocab size = 4000 :

- *Matrice de confusion :*

Predicted Labels True Labels	Not Sexist	Sexist
Not Sexist	557	66
Sexist	150	150

- *Métriques :*

	precision	recall	f1-score	support
Not Sexist	0.79	0.90	0.84	645
Sexist	0.69	0.50	0.58	300
Accuracy			0.77	945
Macro avg	0.74	0.70	0.71	945
Weighted avg	0.76	0.77	0.76	945

c) Analyse des résultats

Les accuracy nous permettent de déterminer à quel point un modèle est performant car il s'agit du pourcentage des prédictions correctes. Lorsque l'on compare toutes les accuracy, on se rend compte que pour la plupart des modèles, l'accuracy est supérieure à 0.70 ce qui signifie que la plupart des modèles sont satisfaisants sans être parfaitement précis, car un modèle parfait aurait une accuracy de 1.

À l'opposé, on constate des résultats un peu moins bon pour le modèle Word2Vec avec Skip Gram et les réseaux de neurones TensorFlow Keras avec oversampling et undersampling. Ces modèles-là ne sont donc certainement pas adaptés aux tâches que l'on souhaite faire ou à nos données initiales.

De plus, comme notre jeu de données n'était pas équilibré, nous avons essayé de calculer des balanced accuracy pour certains de nos modèles. On se rend compte que les balanced accuracy sont plus basses que les accuracy, sauf dans notre modèle Naive Bayes effectué manuellement, pour une raison que l'on ne saurait expliquer.

Également, en se basant uniquement sur les accuracy, on peut en déduire que le meilleur de nos modèles semble être celui du réseau de neurones Tensorflow Keras avec la taille du vocabulaire égale à 400.

Ensuite, lorsque l'on regarde nos matrices de confusion, on se rend compte que dans la majorité des cas, le plus grand nombre d'erreurs correspond au nombre de fois où l'on prédit qu'un tweet est non sexiste alors qu'il était sexiste. Cela peut s'expliquer par le fait que nos données initiales contiennent plus de tweet non-sexistes que de tweet sexistes.

De la même façon, lorsque l'on analyse les métriques de nos modèles, on se rend compte que les valeurs des précisions, des recalls et des f1-score sont souvent plus hautes pour la catégorie "not sexist" que la catégorie "sexist". Ces valeurs nous aident à déterminer à quel point un modèle est performant pour une catégorie en particulier. Par exemple, pour le modèle SVC avec paramètre kernel "rbf", on a une précision de 0.77, un recall de 0.82 et un f1-score de 0.84 pour la catégorie non sexiste alors que pour la catégorie sexiste ces valeurs sont respectivement de 0.49, 0.41 et 0.48. L'écart de performance est donc assez significatif dans cet exemple.

Enfin, lorsque que l'on a recours à de l'oversampling ou à de l'undersampling, on observe que les valeurs des macro average sont identiques ou très proches, ce qui signifie que dans ces cas-là nos données ont effectivement bien été équilibrées.

VI. ANALYSE DES ERREURS

Comme dit précédemment, nous sommes parties d'un jeu de données très complexes et très pauvre. Ce qui fait que nos résultats ne sont malheureusement pas aussi bons que nous le souhaiterions.

En effet, notre corpus est constitué uniquement de tweets. Ce genre de format n'est malheureusement pas réputé pour son grand respect de la langue française. Ainsi, erreurs d'orthographe, de conjugaison et de grammaire s'y trouvent à foison.

De plus, le registre de langue est familier. Les gens écrivent des tweets de la même manière dont ils parleraient dans la vie de tous les jours, ce qui complexifie largement notre tâche de classification.

Par ailleurs, sur plus de 12 000 tweets, à peine plus de 3000 étaient disponibles. C'est bien trop peu pour que notre modèle puisse s'entraîner correctement. Et surtout sur

ces 3000 tweets, les ⅔ étaient catégorisés comme étant non-sexistes. Il n'y a pas simplement un problème de nombre mais aussi d'équilibre !

Ce sont donc toutes ces raisons qui expliquent nos résultats et toutes les erreurs observés !

Toutefois, nous avons tenté du mieux que nous pouvions de pallier ce problème en essayant d'incorporer de l'oversampling et de l'undersampling.

VII. CONCLUSION ET AMÉLIORATIONS

Au cours de ce projet, nous avons dans un premier temps avancé par tâtonnement en ayant recours à des méthodes préexistantes, que nous avons peu à peu améliorées en mêlant parfois différentes techniques (par exemple : oversampling + Naives Bayes), en essayant de reproduire les choses manuellement (Naive Bayes) ou en testant plusieurs paramètres pour un même modèle (par exemple vocab_size pour le réseau de neurones). Nos résultats ne sont peut-être pas à la hauteur du travail réalisé, toutefois, il était intéressant pour nous de toujours essayer d'améliorer un peu plus nos performances. Ainsi, d'après nos expérimentations, le meilleur modèle obtenu est celui du réseau de neurones Tensorflow Keras avec la taille du vocabulaire égale à 400.

Nous avons quelques idées quant à comment améliorer notre projet. Peut-être serait-il préférable de partir de tweets écrits en anglais, étant donné que de nombreux lemmes préfaits et efficaces n'existent qu'en anglais. Et surtout, il nous faudrait partir d'une base de données contenant un grand nombre de tweets, où chaque classe est présente en quantité plus ou moins égale !

SOURCES

(1) Données initiales :
https://github.com/patriChiril/An-Annotated-Corpus-for-Sexism-Detection-in-French-Tweets

(2) Extraction des tweets :
https://developer.twitter.com/en
https://www.askpython.com/python/examples/extracting-tweets-using-twitter-api
https://morioh.com/p/eabbdfdd1f0b

(3) Nettoyages des tweets : ??
https://www.programcreek.com/python/?CodeExample=clean+text

(4) Tutoriel Scikit Learn (Naive Bayes, Grid Search, SVM…) :
https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

(5) Oversampling / Undersampling :
https://towardsdatascience.com/oversampling-and-undersampling-5e2bbaf56dcf

(6) Naive Bayes Classifier:
https://medium.com/analytics-vidhya/naive-bayes-classifier-for-text-classification-556fabaf252b

(7) Tensorflow keras :
https://www.youtube.com/watch?v=15chH6NzhBM
https://github.com/wiqaaas/youtube/blob/master/Deep_Learning_Using_Tensorflow/Text_Classification/Text%20Classification%20with%20Word%20Embeddings.ipynb

(8) Word Embedding :
https://datascientest.com/nlp-word-embedding-word2vec
https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526
https://www.analyticsvidhya.com/blog/2020/08/top-4-sentence-embedding-techniques-using-python/

(9) SVM et Random Forest:
https://analyticsindiamag.com/step-by-step-guide-to-reviews-classification-using-svc-naive-bayes-random-forest/