Classification de Sentiments de Tweets

Jeremy Qin[1] **Santiago Gomez**[2] **Francisco Lopez**[3]

[1]Université de Montréal

https://github.com/Gfecito/STT3795-SentimentAnalysis

1 Contexte

Nous assistons présentement à des avancées majeures dans le domaine de la modélisation de langage et du traitement du langage naturel (NLP). Les progrès de l'apprentissage automatique et de l'apprentissage profond ont notamment permis le développement de modèles de langage de grande envergure, tels que GPT-3 et GPT-4, qui ont révolutionné la façon dont nous abordons la compréhension et la génération de texte de nos jours. Ces modèles ont la capacité d'apprendre à partir d'un grand corpus de données textuelles, ce qui leur permet d'acquérir des connaissances sur le langage et les relations entre les mots.

Ces avancées dans les modèles de langage et le NLP ont un impact significatif sur l'analyse de sentiment, qui est une application importante du NLP. En effet, l'analyse de sentiment joue un aspect crucial dans le monde des affaires et du marketing, en particulier en ce qui concerne l'expérience client. Il consiste principalement à identifier et extraire diverses tonalités et sentiments à partir de données textuelles, tel qu'un commentaire de client ou une critique de produit. Avec la prolifération des réseaux sociaux et la dépendance croissante aux plates-formes numériques pour les commentaires des clients, l'analyse de sentiment est devenue de plus en plus importante.

2 Problématique

La complexité inhérente au traitement du langage naturel a stimulé l'innovation et le développement de nombreuses techniques de prétraitement et de modélisation de données textuelles. Il existe deux approches principales pour la classification de texte: les approches basées sur des règles et les approches basées sur l'apprentissage automatique. Les techniques d'apprentissage automatique telles que la régression logistique, les réseaux de neurones récurrents et les réseaux de neurones profonds ont démontré de solides performances dans divers problèmes de classification de texte.

Cette étude compare les performances de différents modèles d'apprentissage automatique et profond pour la classification de texte. Les modèles évalués comprennent la régression logistique, le perceptron multicouche, les réseaux de neurones convolutifs-recursifs (CNN-RNN) et un ensemble par vote. Cette étude souligne l'importance de l'exploration de différentes techniques de modélisation et de l'utilisation de méthodes d'ensemble pour améliorer les performances et la robustesse des modèles de classification des sentiments du texte.

En résumé, l'objectif de ce projet est donc de découvrir les méthodes de traitement de texte pour le NLP, les utiliser, et utiliser les concepts appris lors du cours avec plusieurs modèles d'apprentissage statistique (et profond) pour finalement en faire une comparaison de nos résultats.

3 Objectifs

Le projet propose les objectifs suivants:

1. Apprendre sur le NLP et nous familiariser avec le domaine.
2. Découvrir différentes méthodes de prétraitement de données textuelles
3. Découvrir différentes techniques de modélisation de données textuelles
4. Appliquer ces techniques pour la classification de sentiments
5. Évaluer et tester les différentes méthodes
6. Comparer les performances des modèles
7. Visualiser nos résultats par différentes métriques

4 Description des données analysées

Pour ce projet, nous nous sommes concentré sur le jeu de données Sentiment140 qui est un ensemble de données volumineux contenant 1,6 million de tweets en anglais et qui ont été étiquetés manuellement par leur polarité de sentiment (positif ou négatif). Ces tweets ont été collectés à partir de la plateforme de médias sociaux Twitter, dans le but de fournir une ressource précieuse pour la recherche en analyse de sentiment. Cette analyse ainsi que la collecte de données a été faite par des étudiants de Stanford avec leur approche décrite plus en détails dans leur papier: Twitter Sentiment Classification using Distant Supervision.

Le jeu de données Sentiment140 est souvent utilisé pour entraîner et tester des modèles de classification de sentiment, ainsi que pour la recherche académique sur l'analyse de sentiment. Il est particulièrement utile pour les chercheurs qui cherchent à développer des modèles de classification de sentiment pour les tweets, en raison de la grande quantité de données disponibles et de leur diversité en termes de contenu et de polarité. Ainsi, notre tâche se résume à une classification binaire des différentes tweets.

Le jeu de données est librement disponible pour une utilisation académique et de recherche, et peut être téléchargé à partir de différents sites Web en ligne comme Kaggle.

5 Méthodologie

Prétraitement des données

Les données textuelles brutes contiennent en grande majorité beaucoup d'informations non pertinentes et peuvent donc nuire à la performance de nos modèles. Les éléments tels que les balises HTML, les caractères spéciaux, les URL, les numéros et la ponctuation peuvent affecter la qualité de l'analyse et même conduire à des résultats erronés. Par conséquent, le nettoyage des données textuelles permet de normaliser le contenu des chaînes de caractères, éliminant les éléments indésirables et laissant uniquement les mots et expressions pertinents pour l'analyse de texte. Le nettoyage des données textuelles est donc une étape cruciale pour améliorer la précision des résultats d'analyse de texte et pour garantir la qualité des données utilisées dans les applications de NLP. Nous commençons donc en premier lieu avec un nettoyage de nos données.

Par la suite, nous utilisons aussi une technique nommée stemming. le stemming est un processus qui consiste à réduire un mot à sa racine, également appelée lemme, en supprimant les suffixes et les préfixes. Il permet notamment de normaliser les mots et de réduire la dimensionnalité de l'espace des caractéristiques en regroupant les formes fléchies d'un même mot sous une seule forme. Le stemming est une technique très populaire et utile pour l'analyse de sentiments ainsi que le NLP en général. En effet, l'analyse de sentiments implique souvent la recherche et la classification des termes qui expriment des émotions ou des sentiments. Le stemming permet de regrouper les variantes d'un même mot sous une seule forme, ce qui facilite l'identification et la classification de ces termes. En utilisant le stemming dans l'analyse de sentiments, il est donc possible de détecter plus facilement les termes qui expriment des émotions et des sentiments, ce qui permet d'améliorer la précision de l'analyse.

TfidfVectorizer et Tokenizer

Pour que nous puissions donner à notre modèle des données sur lesquelles il est capable d'apprendre et de s'entraîner, nous devons encore passer par une étape de prétraitement. Dépendemment du modèle, nous utilisons différentes méthodes: soit Tokenizer soit TfidVectorizer. Nous présenterons tout d'abord la méthode intitulée TfidtfVectorizer.

TfidfVectorizer (Term Frequency Inverse Document Frequency) est un outil de traitement de langage naturel qui permet de transformer les données textuelles en vecteurs numériques afin de pouvoir les donner en entrée à des modèles d'apprentissage machine. TfidfVectorizer calcule un score pour chaque mot en se basant sur deux critères : la fréquence du mot dans le document et sa fréquence dans l'ensemble des documents. Les mots qui apparaissent fréquemment dans un document, mais rarement dans les autres documents, obtiennent un score élevé, ce qui indique qu'ils sont des caractéristiques importantes du document. À l'inverse, les mots qui apparaissent fréquemment dans tous les documents obtiennent un score plus faible, car ils n'apportent pas d'informations discriminantes pour la classification. Nous pouvons maintenant regarder plus en détails les formules derrières cette mesure statistique.

Soit N : nombre de documents (tweets),
d : document quelconque,
D : collection de tous les documents,
w : mot quelconque d'un document
On définit la fréquence de terme (term frequency) comme étant

$$tf(w,d) = log(1 + f(w,d))$$

$f(w,d)$: fréquence du mot w dans le document d

On définit aussi l'inverse de fréquence de terme comme étant

$$idf(w,D) = log(\frac{N}{f(w,D)})$$

Avec ces définitions, nous pouvons écrire le score TF-IDF comme étant

$$tfidf(w,d,D) = tf(w,d) * idf(w,D)$$

Ainsi, nous nous retrouvons avec des représentations adéquates de nos données textuelles sous forme de vecteurs ce qui nous permettra par la suite d'appliquer des algorithmes de classification pour la détection de sentiments.

Ensuite, nous avons aussi utilisé une autre technique de prétraitement nommé Tokenizer pour notre modèle de réseaux de neurones récurrents (RNN) dont nous aborderons après. En effet, pour utiliser un modèle RNN pour le traitement de texte, il est nécessaire de fournir au modèle des séquences de mots ou de caractères qui ont été préalablement divisées en "tokens" ou "jetons". Le processus de tokenization est donc nécessaire pour préparer les données de texte pour l'entraînement et la prédiction. La tokenization est un processus clé en traitement automatique du langage naturel qui consiste à diviser un texte en une séquence de "tokens", qui sont des unités discrètes de texte telles que des mots, des symboles de ponctuation et des chiffres. Elle peut être réalisée en utilisant des règles simples basées sur l'espacement et la ponctuation, ou en utilisant des algorithmes plus sophistiqués basés sur l'apprentissage automatique. La tokenization permet aussi de normaliser le texte en réduisant les variations dans la façon dont les gens écrivent et formatent leur texte. Cela facilite le traitement automatique du texte par les machines en créant une représentation numérique du texte sous forme de séquences de tokens, qui peuvent être encodées sous forme de vecteurs numériques.

Modeles utilisées

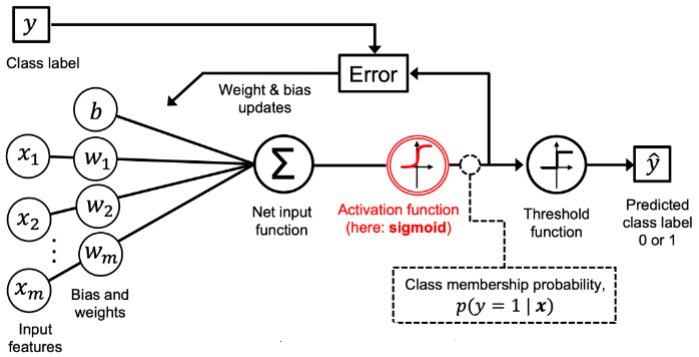
Pour modéliser et prédire les données, nous avons utilisé trois approches : la régression logistique, les réseaux de neurones multicouches (MLP) et les réseaux de neurones convolutifs récurrents (CNN-RNN).

Regression Logistique. La régression logistique est un modèle linéaire simple qui fonctionne bien pour les tâches de classification binaire. En effet, elle estime la probabilité qu'un évènement se produise. Comme on estime une probabilité, la valeur est bornée entre 0 et 1. La régression logistique applique alors une transformation logit sur la cote, qui est la probabilité de succès divisée par la probabilité de défaillance. Nous aurions donc les formules suivantes:

$$Logit(\pi) = \frac{1}{1 + exp(-\pi)}$$

$$ln(\pi) = \beta_0 + \beta_1 X_1 + ... + \beta_k X_k$$

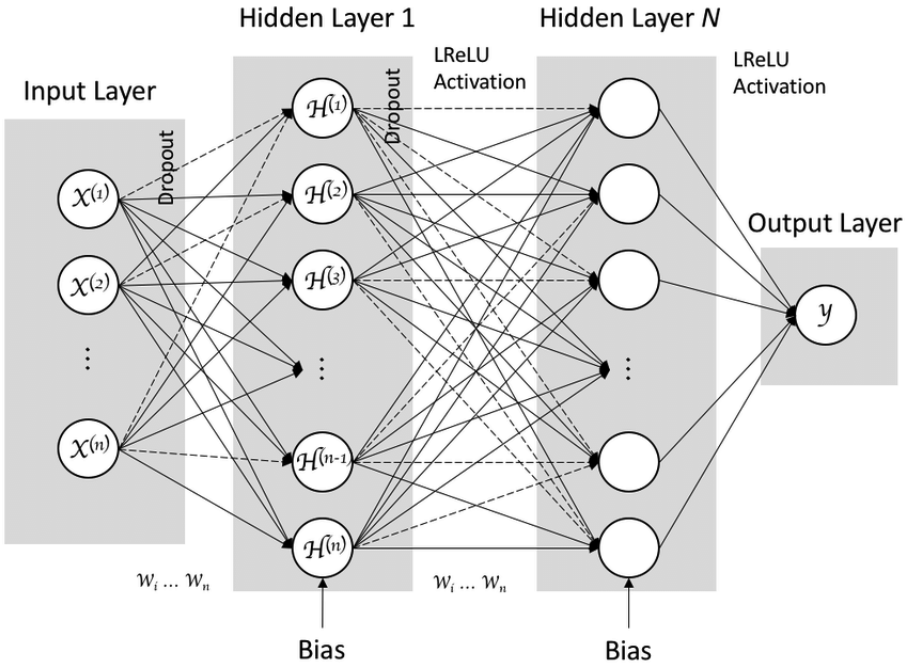
Afin de mieux visualiser la régression logistique, la figure ci-dessous donne un aperçu de comment elle fonctionne .



Architecture Regression Logistique
(pris du article Logistic Regression Concepts, Python Example par Ajitesh Kumar)

Pour pouvoir performer la classification binaire, un seuil est mis à 0.5 ce qui fait que les probabilités inférieures à celui-ci est classées dans une classe et les probabilités supérieures sont classées dans une autre.

Multilayer Perceptron. Les MLP sont des réseaux de neurones profonds capables de modéliser des relations non linéaires complexes dans les données. En effet, il est constitué notamment de plusieurs couches neuronales dont une couche d'entrée, une couche de sortie et plusieurs couches cachées d'où vient le nom d'apprentissage profond. Chaque neurone est connecté aux autres neurones de la prochaine couche et ces connections possèdent un poids et un seuil associées. Si la sortie d'un neurone est supérieure au seuil, le neurone est activé et l'information passe à la prochaine couche. Dans le cas contraire, l'information n'est pas envoyée. Plusieurs fonctions d'activations existent, et nous avons choisi la fonction ReLU qui est couramment utilisé à travers multiples architectures de réseau de neurones. C'est notamment grâce aux fonctions d'activations que le réseau est capable d'apprendre des relations non-linéaires dans nos données. Une manière simple de voir un réseau de neurones est de le voir comme étant plusieurs régressions linéaires composées. L'apprentissage du réseau se base sur le concept de rétropropagation et de descente de gradients. Nous omettons les détails concernant la théorie derrière l'optimisation comme ce n'est pas la propos principal de ce projet. Pour nos expériences, nous avons testé des réseaux avec une couche d'entrée avec 64 neurones, une cachée avec 32 neurones et une de sortie avec 2 neurones. L'optimization a été fait avec l'algorithme d'Adam qui se base sur la descente de gradient.



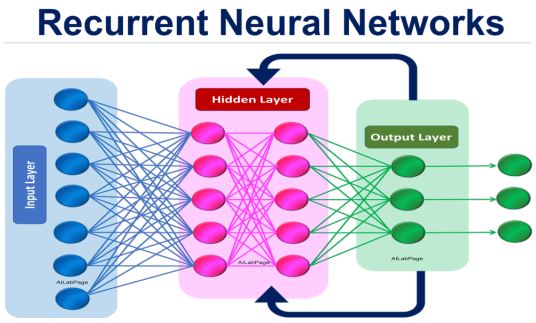
Architecture MLP
(MLP deep learning architecture par Ryan Heartfield)

Convolutional-Recurrent Neural Network. Un RNN (Réseau de Neurones Récurrent) est un type d'architecture de réseau neuronal conçue pour traiter des données séquentielles, telles que des séries temporelles ou du texte en langage naturel. Contrairement aux réseaux neuronaux à propagation directe, qui traitent les données en une seule passe, les RNN ont une boucle de rétroaction qui leur permet de transmettre des informations d'une étape de la séquence à la suivante.

La caractéristique clé d'un RNN est qu'il possède un état caché, qui est mis à jour à chaque étape de la séquence en fonction de l'entrée actuelle et de l'état caché précédent. L'état caché agit comme une mémoire des entrées précédentes, permettant au RNN de capturer les dépendances entre les éléments de la séquence, ce qui est puissant avec données de relations séquentielles tel que les données textuels.

Un type courant de RNN est le réseau LSTM (Long Short-Term Memory), qui est conçu pour résoudre le problème des gradients qui disparaissent et qui peuvent survenir dans les RNN traditionnels. Les LSTM utilisent un mécanisme de mise à jour plus complexe pour l'état caché qui leur permet de conserver des informations sur de plus longues périodes de temps.

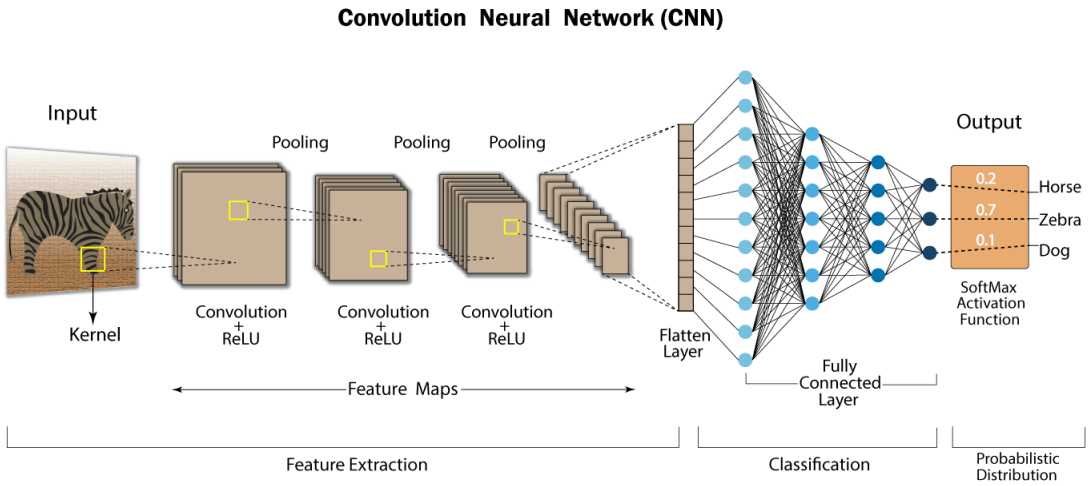
Les RNN sont utilisés dans de nombreuses applications, notamment la reconnaissance vocale, le traitement du langage naturel, la légende d'images et la génération de musique, entre autres.



Architecture RNN
(pris du article RNN vs 1D Convolutional Networks par Andrei Moiceanu)

Un CNN (Convolutional Neural Network) est un type de réseau neuronal qui est particulièrement adapté pour le traitement de données en deux dimensions, comme des images ou des vidéos. Les CNN sont conçus pour capturer les caractéristiques visuelles des données d'entrée en utilisant des opérations de convolution. Toutefois, les propriétés avantageux de la convolution n'est pas seulement restreinte à des images ou des vidéos. Par simplicité, nous aborderons comment la convolution s'applique pour des images.

Les opérations de convolution impliquent l'application d'un filtre à une image pour produire une carte de caractéristiques, qui met en évidence des motifs spécifiques tels que des bords, des coins, ou des textures. Cela consiste principalement à estimer une valeur à un certain point en faisant une somme pondérée des points voisins tout en donnant plus de poids à des points plus proches.



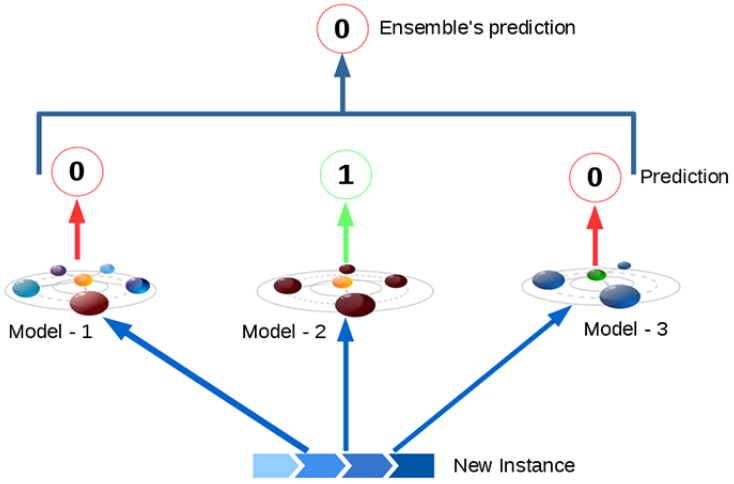
Architecture CNN
(pris du article Basics of CNN in Deep Learning par Debasish Kalita)

Les CNN sont généralement utilisés pour la vision par ordinateur mais peuvent également être appliqués au texte. Nous avons utilisé des convolutions 1D qui glissent une fenêtre sur les séquences de texte pour apprendre des motifs locaux.

Comme ce réseau de neurones est beaucoup plus gros, nous avons aussi utilisé des couches de Dropout afin de régulariser le modèle pour éviter du surajustement (overfitting).

Voting Ensemble

Comme dernier technique nous avons finalement combiné les trois modèles précédents en un ensemble par vote majoritaire. Chaque modèle contribue à un vote de manière équivalente car ils ont des performances similaires. L'ensemble final a prédit les étiquettes qui ont reçu le plus de votes de la part des modèles individuels.



Architecture du méthode du Voting Ensemble
(pris du article Heterogeneous Ensemble Learning (Hard voting / Soft voting))

Le principe du Voting Ensemble s'inspire du phénomène de "wisdom of the crowds", où on estime qu'un ensemble des agents intelligents engendrerait des meilleurs estimations que les individus eux mêmes, puisque leurs biais pourront se annuler entre eux.

6 Résultats et analyse

Voici les résultats obtenus

- 0 et 1 représentent les classes des sentiments négatives et positives respectivement.
- Dans ce cas ci, les sections du macro avg ET du weighted avg sont toutes les deux équivalents à la métrique moyenne des deux classes.
- Support indique la quantité des datapoints dont nos tests s'exécutent.
- Précision est une métrique qui nous indique la fréquence dont un point d'une classe donnée est bel et bien classifié comme étant dans cette classe.
- Recall est une métrique qui nous indique la fréquence dont un point n'appartenant pas a une classe donnée est bel et bien classifié comme étant dans une autre classe.

Régression Logistique

	Precision	Recall	f1-score	Support
0	0.80	0.78	0.79	39896
1	0.79	0.81	0.80	40104
accuracy			0.79	80000
macro avg	0.79	0.79	0.79	80000
weighted avg	0.79	0.79	0.79	80000

Multilayer Perceptron

	Precision	Recall	f1-score	Support
0	0.79	0.81	0.80	39896
1	0.80	0.79	0.80	40104
accuracy			0.80	80000
macro avg	0.80	0.80	0.80	80000
weighted avg	0.80	0.80	0.80	80000

CNN-RNN

	Precision	Recall	f1-score	Support
0	0.81	0.84	0.82	39896
1	0.83	0.81	0.82	40104
accuracy			0.82	80000
macro avg	0.82	0.82	0.82	80000
weighted avg	0.82	0.82	0.82	80000

Voting Ensemble

	Precision	Recall	f1-score	Support
0	0.81	0.82	0.82	39896
1	0.82	0.81	0.82	40104
accuracy			0.82	80000
macro avg	0.82	0.82	0.82	80000
weighted avg	0.82	0.82	0.82	80000

D'après le résultats obtenus pour chaque modèle on peut constater:

Régression logistique: Cette méthode a la précision, le rappel et les scores F1 les plus bas parmi les quatre méthodes. Le score F1 moyen pondéré est de 0,79. Ceci suggère que la régression logistique ne fonctionne pas aussi bien que les autres méthodes pour cet ensemble de données.

Perceptron multicouche: Cette méthode a de légèrement meilleures performances que la régression logistique avec un score F1 moyen pondéré de 0,80. La précision, le rappel et les scores F1 sont un peu plus équilibrés entre les deux classes. Ce modèle de réseau de neurones fonctionne donc mieux mais pas aussi bien que les deux autres méthodes.

CNN-RNN: Cette méthode obtient les meilleures performances avec un score F1 moyen pondéré de 0,82. La précision, le rappel et les scores F1 sont les plus élevés pour cette méthode, en particulier pour la classe 0. Le modèle d'apprentissage profond avec des couches CNN et RNN est donc le plus adapté à cet ensemble de données.

Ensemble par vote: Cette méthode d'ensemble obtient également un score F1 moyen pondéré élevé de 0,82. La précision, le rappel et les scores F1 sont très similaires au modèle CNN-RNN. Ceci suggère que l'agrégation des prédictions de plusieurs classifieurs (CNN, RNN, régression logistique, MLP) conduit à une amélioration des performances équivalente au meilleur modèle individuel (CNN-RNN).

En résumé, les modèles d'apprentissage profond (CNN-RNN et ensemble par vote) ont des performances un peu meilleures que les modèles de régression logistique et MLP pour cet ensemble de données. Toutefois, il est difficile d'assumer que cette différence est significative. Pour pouvoir être certain de cela, il va falloir utiliser des méthodes de validation comme du K-fold cross-validation ou LOOCV. Toutefois, cela ne sera pas abordé dans ce projet, mais serait intéressant comme travail futur. Le modèle CNN-RNN obtient les meilleures performances individuellement, mais l'ensemble par vote égale ses performances, ce qui indique que l'ensemble peut atteindre une précision similaire avec plus de stabilité et de robustesse que seulement utiliser le CNN-RNN, ce qui est rassurant pourvu qu'on s'attend que les réseaux RNN soit mieux adaptés à résoudre les problèmes du NLP et on voit également que le vote d'ensemble est tout à fait aussi capable que notre modèle le plus compètent. Bien que notre étude comportait des limites, elle souligne l'importance de l'exploration de différents modèles et techniques d'ensemble pour améliorer les performances en classification de texte.

7 Conclusion

Dans ce projet, nous avons étudié et comparé les performances de trois modèles d'apprentissage automatique - la régression logistique, les réseaux de neurones récurrents (RNN) et les réseaux de neurones profonds (CNN) - ainsi qu'une méthode d'ensemble par vote pour la classification de texte. Les résultats ont montré que les modèles d'apprentissage profond, en particulier le CNN-RNN et l'ensemble par vote, surpassent les modèles d'apprentissage automatique traditionnels tels que la régression logistique et le perceptron multicouche (MLP).

Les limitations de notre étude incluent l'utilisation d'un ensemble de données spécifique pour évaluer les performances des différentes méthodes. Il est possible que les résultats varient en fonction de l'ensemble de données utilisé, de la qualité des données et de la représentation des différentes classes. De plus, nous avons examiné un nombre limité de modèles d'apprentissage automatique et d'apprentissage profond, laissant de côté d'autres modèles potentiellement

performants comme les forêts aléatoires, les machines à vecteurs de support (SVM) ou les transformateurs.

Pour améliorer cette étude à l'avenir, il serait intéressant d'explorer d'autres modèles de classification, tels que les forêts aléatoires qui donnent d'habitude des résultats semblables et aussi est facile a interpréter. De plus, il serait pertinent d'effectuer des expériences supplémentaires avec différents ensembles de données et des pré-traitements de données variés pour évaluer la robustesse et la généralisation des modèles. Il est également possible d'explorer des techniques d'optimisation des hyperparamètres pour améliorer les performances des modèles existants.

En conclusion, notre étude a révélé que les modèles d'apprentissage profond, en particulier le CNN-RNN et l'ensemble par vote, surpassent les modèles d'apprentissage automatique traditionnels pour la classification de texte. Le modèle CNN-RNN obtient les meilleures performances individuellement, mais l'ensemble par vote égale ses performances, ce qui indique que l'ensemble peut atteindre une précision similaire avec plus de stabilité et de robustesse que seulement utiliser le CNN-RNN. Cela souligne l'importance de l'exploration de différentes techniques de modélisation et l'utilisation de méthodes d'ensemble pour améliorer les performances et la robustesse des modèles de classification de texte.

8 Contributions des membres

La contribution de nos membres s'est faite de la façon suivante:

- Jeremy Qin: gestion principale du projet et développement; chargé de l'exécution des expériences et développeur principal du code. Contribué et dirigé le rapport. Recherche des méthodes de pré-traitement, modèles, et des datasets candidats.
- Santiago Gomez: développeur auxiliaire du code et écriture du rapport. Recherche des méthodes de pré-traitement, modèles, et des datasets candidats.
- Francisco Lopez: développeur auxiliaire du code et écriture du rapport. Recherche des méthodes de pré-traitement, modèles, et des datasets candidats.

References

A Figures

KumarI, A. (2023, January 26). Logistic Regression Concepts, python example. Data Analytics. Retrieved April 27, 2023, from https://vitalflux.com/python-train-model-logistic-regression/

Heartfield, R. (2017). MLP deep learning architecture | scientific diagram - researchgate. researchgate. Retrieved April 27, 2023, from https://www.researchgate.net/figure/MLP-deep-learning-architecture_fig5_321341597

Moiceanu, A. (2022, June 29). Recurrent neural networks vs 1D convolutional networks. Medium. Retrieved April 27, 2023, from https://medium.com/mlearning-ai/recurrent-neural-networks-vs-1d-convolutional-networks-5ac7b4f68ca9

Kalita, D. (2022, March 22). Basics of CNN in Deep Learning. Analytics Vidhya. Retrieved April 27, 2023, from https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/

A. (2019, June 10). Heterogeneous ensemble learning (hard voting / soft voting). datajango. Retrieved April 27, 2023, from https://www.datajango.com/heterogeneous-ensemble-learning-hard-voting-soft-voting/

B Dataset

Anova, K. (2017, September 13). Sentiment140 dataset with 1.6 million tweets. Kaggle. Retrieved April 27, 2023, from https://www.kaggle.com/datasets/kazanova/sentiment140