Prédiction de sentiments par apprentissage automatique

Augustin BAR

Table des matières

[I. Introduction 2](#_Toc110516709)

[A. Problématique 2](#_Toc110516710)

[B. Analyse du jeu de données 2](#_Toc110516711)

[C. Approche et méthode de travail 2](#_Toc110516712)

[II. Approche simple 3](#_Toc110516713)

[A. Traitement du texte 3](#_Toc110516714)

[B. Vectorisation 3](#_Toc110516715)

[C. Prédiction par différents modèles 3](#_Toc110516716)

[D. Améliorations pour la production 3](#_Toc110516717)

[III. Approche complexe 4](#_Toc110516718)

[A. Traitement du texte 4](#_Toc110516719)

[B. Plongement lexical 4](#_Toc110516720)

[C. Création d’un réseau de neurone personnalisé 4](#_Toc110516721)

[D. Améliorations pour la production du produit 4](#_Toc110516722)

[IV. Pour aller plus loin 5](#_Toc110516723)

[A. Implémentation d’un transformer 5](#_Toc110516724)

[B. Possibilité d’amélioration 5](#_Toc110516725)

[V. Conclusion 6](#_Toc110516726)

[A. Cas d’usage 6](#_Toc110516727)

[B. Améliorations possibles 6](#_Toc110516728)

# Introduction

## Problématique

Afin d’améliorer son image, en particulier sur les réseaux sociaux, notre client nous a demandé d’élaborer un système leur permettant de prédire la réaction du public lors de leurs publication.

Pour cela, nous avons donc voulu concevoir un système auto-hébergé qui, grâce à l’intelligence artificielle et à l’apprentissage automatique, leur permettrait de prédire le sentiment associé à un tweet via un appel API.

## Analyse du jeu de données

Le client ne nous ayant pas fourni de jeu de données. Comme il tweetera en anglais, nous avons donc cherché un jeu de données de tweets anglais labellisé selon leurs sentiments.

Nous avons finalement choisi d’utiliser un jeu de données Open Source et en libre accès à [cette adresse](https://www.kaggle.com/datasets/kazanova/sentiment140).

On remarque après étude que ce jeu de donnée n’a que deux classes (contrairement aux 3 annoncées) mais que les deux restantes sont équilibrées. Finalement, ce jeu de données correspond bien à la tâche que nous souhaitons réaliser.

## Approche et méthode de travail

Afin de montrer au client les différentes solutions disponibles ainsi que leurs avantages et inconvénient, nous avons proposé de créer 3 preuves de concept.

Bien que seule une de ces 3 différentes approches soit hébergée et disponible à l’essai, les 3 solutions seront implémentées dans un notebook accessible au client et documenté pour des initiées.

La première approche choisie sera la plus simple, basée sur une technique de sac de mots pour faire plongement de mots puis une comparaison entre 3 différents estimateurs statistiques simples qui détermineront le sentiment associé au tweet.

La seconde approche mettra en avant un réseau de neurone personnalisé dont les premières couches seront empruntées aux célèbres modèles de type Word2Vec auquel nous associerons nos couches de neurones personnalisées afin de faire une prédiction.

Enfin, pour proposer une approche à la pointe de la recherche actuelle, nous proposerons l’utilisation d’un modèle de type BERT, développé par Google en 2018.

Afin de mesurer la performance des modèles, nous utiliserons la métrique AUC (aire sous la fonction d’efficacité du récepteur) dont vous pouvez trouver les détails [ici](https://www.wikiwand.com/fr/Courbe_ROC).

En tant que preuves de concept, ces trois approches ne bénéficieront pas de fine-tuning, afin de raccourcir le temps de conception et de laisser une marge de progression lors de la production du produit final.

# Approche simple

## Traitement du texte

Lors de l’utilisation de phrases et mots d’usage en apprentissage automatique, il est très souvent conseillé d’appliquer des transformations de ces mots afin de normaliser et épurer le jeu de donnée.

Par exemple, certains mots comme « un » ou « la » en français n’apportent que très peu de sens à la phrase et nécessitent d’être retirés afin de ne pas surcharger les données d’entrée. Attention cependant, pour certaines tâches comme la nôtre, il serait une grosse erreur de retirer les mots « ne » ou « pas » ou autre négation car elles apportent une négation à la phrase qui change drastiquement son sens.

Après avoir retiré les mots inutiles, nous choisissons d’appliquer une technique d’uniformisation des mots. En effet, pour notre tâche, garder différentes formes comme « aimer », « aime » ou « aimons » n’apporte que peu d’intérêt et nous choisissons de ne relier toutes ses former qu’à leur forme « originelle », ici « aimer ».

## Vectorisation

Les modèles mathématiques n’étant pas capables de d’utiliser des mots, nous devons transformer les tweets (listes de mots) en liste de nombres, c’est ce que l’on appelle le plongement de mots.

Pour cela, il existe plusieurs solutions, mais comme nous sommes dans une approche simple, nous allons utiliser un vectoriseur qui n’est entrainé que sur notre propre jeu de données. Le principe est simple, le vectoriseur analyse le jeu de données et créé un dictionnaire de tous les mots auquel il associe un numéro. Enfin, dans chaque phrase, il remplace le mot par le numéro associé.

## Prédiction par différents modèles

Les données étant prêtes à être utilisées, vient le problème du choix du modèle.

Nous avons décidé d’implémenter 3 modèles simples différents pour comparer leurs performances.

Nous ne détaillerons pas le fonctionnement des différents modèles dans ce document mais vous pouvez retrouver leur fonctionnement dans les liens associés.

1. Régression logistique : [Documentation](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
2. Classification naïve bayésienne : [Documentation](https://scikit-learn.org/stable/modules/naive_bayes.html#bernoulli-naive-bayes)
3. Machine à vecteur de support : [Documentation](https://scikit-learn.org/stable/modules/svm.html#svm-classification)

Finalement, les résultats ont été plutôt bons, avec une AUC de 0.70 pour chacun des modèles.

## Améliorations pour la production

Si ce prototype est choisi par le client, il est clair qu’un entrainement plus poussé et des techniques de fine-tuning devront être mises en œuvre pour améliorer la performance des modèles.

Aussi, il serait plus utile d’utiliser un vectoriseur déjà entrainé sur un lexique beaucoup plus complet afin de ne pas manquer de lexique pour les mots qui vont être proposés par l’API.

# Approche complexe

Pour cette approche complexe, nous avons choisi de prendre des technologies plus poussées pour le plongement de mots et pour la prédiction, et de les concaténer pour en faire un réseau de neurones.

## Traitement du texte

Le traitement du texte est le même que pour l’approche simple.

## Plongement lexical

Pour le plongement de mots, nous allons utiliser le modèle appelé glove25 que vous pourrez retrouver sur [ce site](https://nlp.stanford.edu/projects/glove/). Ce vectoriseur a été entrainé sur plus de 2 milliards de tweets. Glove 25 signifie que chaque mot est représenté par 25 valeurs (donc dans un espace à 25 dimensions), ce qui permet au modèle de beaucoup mieux comprendre les similitudes entre les mots.

## Création d’un réseau de neurone personnalisé

Enfin, pour la prédiction, nous avons utilisé des couches de neurones de convolution et dense à la fin. Cela permet à notre réseau d’apprendre et nous donner une prédiction à la fin si le tweet a un sentiment bon ou mauvais.

Finalement, l’entrainement se passe sans problèmes d’apprentissage et le résultat après 10 cycles d’entrainement est très prometteur avec une AUC de 0.77.

## Améliorations pour la production du produit

Lors de la production du produit, ci ce prototype est choisi, un entrainement plus long ainsi qu’un entrainement très léger sur le vectoriseur pourrait être une bonne idée. Attention tout de même aux problèmes de surapprentissage.

# Pour aller plus loin

## Implémentation d’un transformer

Une idée proposée par le client était la mise en place d’un transformer. Pour le traitement du langage, la recherche avance très vite et en 2022, le meilleur reste encore d’utiliser des semi-transformers comme BERT.

C’est ce que nous avons voulu faire en essayant [DistilBERT](https://huggingface.co/docs/transformers/model_doc/distilbert) (un réseau plus léger et rapide copié sur BERT).

Les résultats sont très impressionnants et sans entrainement, nous arrivons à trouver une AUC de 0.78.

## Possibilité d’amélioration

Nul doute qu’avec plus d’entrainement et de fine tuning, on peut arriver à améliorer encore les performances de ce modèle.

# Conclusion

## Cas d’usage

## Améliorations possibles