

# L'impact de la tokenisation sur la détection de sentiments

Hugo Laface

*Maîtrise en Technologies de l'Information  
École de Technologie Supérieure  
Montréal, QC, Canada  
hugo.laface.1@ens.etsmtl.ca*

Raphaël Largeau

*Maîtrise en Technologies de l'Information  
École de Technologie Supérieure  
Montréal, QC, Canada  
raphael.largeau.1@ens.etsmtl.ca*

**Résumé**—L'analyse de sentiments à partir de tweets est une tâche complexe attribuée à de nombreux modèles de Natural Language Processing (Naïve Bayes, k-Nearest Neighbor, Support Vector Machine, Artificial Neural Networks etc...). Dans le cadre de l'étude de ces modèles nous avons comparé différentes paramétrisations de tokenisation et de vectorisation des données. L'objectif est de mettre en valeur l'influence du type de tokenisation ainsi que celle du dictionnaire employé dans le but d'en ressortir la préparation de données optimale pour de l'analyse de sentiments de tweets en anglais. Il s'est avéré que la volonté de maximiser tous les paramètres (comme le nombre de vecteurs ou le grammage) rendait les modèles moins précis. La complexité de la langue et en particulier celle du “langage Twitter” est également un frein à la recherche de performances optimales. Nous avons ainsi pu trouver une configuration des paramètres pour maximiser les performances. Finalement, notre modèle léger, couplé à un pré-processing idéal, reste moins précis dans la détection de sentiments que l'est un modèle profond spécialement entraîné pour cette tâche (BERT). Cependant les performances finales sont tout de même bonnes et le modèle offre un fervent compromis alliant précision et rapidité de prédiction.

**Mots-clés**—Tweets, tokenisation, détection de sentiments, NLP, apprentissage machine

## I. INTRODUCTION

L'analyse des sentiments dans les tweets a de plus en plus d'applications pour un grand nombre d'organisations et d'entreprises. Cela permet, par exemple, d'étudier la satisfaction d'un client ou d'évaluer l'opinion publique sur un sujet. Elle peut même permettre de prévoir l'évolution d'une pandémie en observant les émotions majoritaires sur un sujet précis selon les différentes régions du monde. Cependant, avec les technologies actuelles, une étude précise des sentiments sur de nombreux tweets demande une importante puissance de calcul.

C'est pourquoi, nous souhaitons étudier l'impact de différents tokenizers sur la performance de modèles d'analyse textuelle de sentiments. Notre travail s'est axé sur l'étude de différents paramètres de préparation et de tokenisation des tweets. Ensuite, pour chaque ensemble de ces paramètres, nous avons entraîné un modèle simple (machine à vecteurs de sup-

port). Puis, nous avons comparé ces modèles entre eux selon les ensembles de paramètres et avec un modèle Bert pré-entraîné pour la détection d'émotions au sein de tweets. Finalement, nous répondrons à la question : quel impact le prétraitement et plus particulièrement la tokenization ont ils sur la détection de sentiments dans des données textuelles ?

## II. REVUE DE LITTÉRATURE

[2] [1] [5] La détection de sentiments est un sujet de recherche très répandu et qui témoigne d'un intérêt grandissant depuis la numérisation de la société. Divers travaux sont donc menés dans l'optique d'améliorer cette détection. Un intérêt tout particulier est porté à la phase de pré-processing car jugée moins gourmande en ressource que le processing lui-même. La plupart des recherches menées dans ce cadre visent à comparer des tokenizers existant afin d'en déduire le plus performant. Peu d'entre elles s'intéressent concrètement au fonctionnement profond et à l'impact des (hyper)paramètres sur la tokenisation. On note par exemple les travaux de Sanghyun Choo et Wonjoon Kim dans lesquels ils [1] comparent deux tokenizers (Mecan-ko et SentencePiece) pour de l'analyse de sentiments contenus dans des avis d'achats coréens. Leur approche est différente de la nôtre dans le sens où il recherche le plus performant en association avec plus algorithme de classification (Naïve Bayes, k-Nearest Neighbor, Support Vector Machine, Artificial Neural Networks, et Long Short-Term Memory Recurrent Neural Networks). De plus, ils utilisent des tokenizers complets et disposant d'algorithmes et de composants particuliers, couplé à l'usage de 5 modèles différents leurs résultats sont orientés vers l'ultra-performance plutôt qu'un compromis performance-légereté. Dans [5], différents outils de tokenisation sont comparés, dont NLTK que nous utilisons dans notre recherche, pour en analyser les performances. Un travail notable, mais une fois de plus, qui ne s'attarde pas sur le fonctionnement profond des outils. Il

existe toute fois des travaux menés sur certains paramètres spécifiques, comme par le Dr. Vijayarani Mohan dans [2] qui mène une étude complète sur les émojis au sein de la tokenization. Travail selon lequel les émojis améliore cette détection de sentiments et pourrait avec un impact sur la compréhension d'émotions plus complexes comme le sarcasme, l'ironie ou l'humour noir.

### III. OBJECTIFS

Au cours de ce projet, nous nous sommes fixé l'objectif d'étudier un maximum de paramètres de préparation et de tokenisation des tweets, puis d'en déduire leurs influences et leurs importances dans la détection de sentiments. Nous avons ensuite divisé cet objectif principal en trois étapes. La première étape consiste à pouvoir appliquer différents paramètres de pré-traitement et de tokenisation sur nos données. Les paramètres que nous souhaitons étudier sont notamment : les n-grams, les stopwords, la présence ou non d'émojis, la méthode de vectorisation des tokens et la taille du vecteur d'encodage. La deuxième étape vise à entraîner un modèle (machine à vecteur de support) avec différentes configuration de paramètres avec nos données afin d'en ressortir des résultats de précision et de performance à comparer. Enfin, la troisième étape concerne l'étude des prédictions d'un modèle plus puissance d'apprentissage profond pré-entraîné (Bert) sur nos données en comparaison des résultats obtenus à l'étape précédente.

### IV. MATÉRIEL

#### A. DONNÉES

Nous avons choisi nos données parmi les datasets de tweets disponible sur Kaggle. Puisque notre projet consiste à pré-traiter au mieux des tweets, nous avions besoin de tweets dans leur état le plus brut. Par ailleurs, nous voulions que ces tweets soient tous annotés avec des émotions plus complexes que "positif" ou "négatif". Notre dernière exigence était d'avoir un grand nombre de tweets pour pouvoir entraîner de manière fiable un modèle.

À partir de ces contraintes, nous avons choisi le dataset [6]. Il est composé de 69298 tweets complètement bruts sur divers sujets, avec parfois des liens, des noms d'utilisateurs et des caractères spéciaux comme des émojis. Ces tweets sont tous rédigés en anglais, ce qui simplifie notre travail. De plus, chaque tweet est annoté par une émotion parmi 7 différentes. Les émotions possibles avec leur nombre d'instances respectif dans le dataset sont : "neutral" (24659), "sad-

ness" (13029), "Joy" (10076), "Love" (7402), "Anger" (6567), "Surprise" (3933), "Fear" (3632). Ce dataset répond donc bien à nos exigences. Cependant, la proportion de représentation des classes n'est pas équitable, ce qui nous amènera à faire quelques ajustements dans la préparation des données.

#### B. OUTILS

Pour réaliser notre projet, nous avons eu recours à python en raison de la grande variété de bibliothèques que ce langage propose, notamment dans le domaine du traitement des données et de l'apprentissage machine. De plus ce langage nous permet d'utiliser à la fois notre modèle léger (machine à vecteur de support) et le modèle Bert, dans des conditions similaires, ce qui permet de mieux les comparer.

Toute notre phase de préparation et de tokenisation utilise les bibliothèques nltk et transformers. En effet, ces bibliothèques fournissent toutes les fonctions nécessaires au nettoyage et à la tokenisation de tweets selon tous les paramètres que nous visons d'étudier.

Pour notre modèle, nous avons utilisé une machine à vecteur de support fournie par scikitlearn (LinearSVC), cela correspond parfaitement à nos besoins puisque cette bibliothèque est optimisée et permet d'entraîner rapidement de nombreux modèles sur un CPU.

L'implémentation de Bert que nous avons utilisé, quant à elle, a été implémentée avec pytorch et provient de [4]. Nous avons utilisé ce modèle sur CPU pour que les performances soient comparables à celles du modèle précédent.

Finalement, pour tout l'affichage de nos résultats, nous avons fait le choix d'utiliser matplotlib pour l'ensemble des possibilités qu'elle offre, ce qui nous permet de mieux mettre en valeur nos résultats.

### V. MÉTHODES

#### A. PRÉ-TRAITEMENT ET TOKENISATION

Comme nous l'avons précisé plus tôt, nos données sont des tweets bruts sur divers sujets. Avant la tokenisation, il convient donc de les nettoyer. Pour faire cela, nous avons d'abord passé tous les tweets en minuscules. Ensuite, nous avons retiré les liens et les noms d'utilisateurs des tweets en partant de l'hypothèse qu'ils ne portent pas de signification facilement exploitable. Pour simplifier les tweets, nous avons retiré tous les numéros et caractères spéciaux, sauf les émojis que nous avons seulement regroupés par catégories en les remplaçant par des mot-clés et sur lesquels nous reviendrons plus tard. Nous avons aussi re-

tiré la ponctuation pour simplifier le problème, mais avec plus de temps, nous aurions voulu rajouter des tokens particulier pour certaines formes de ponctuation comme les exclamations et les interrogations.

Une fois la première étape de nettoyage préliminaire effectuée sur tous les tweets, nous avons commencé à appliquer différents paramètres à leur pré-traitement et à leur tokenisation. Ces paramètres sont : la prise en compte ou non des émojis, la suppression ou non des stopwords dans les tweets, les n-grams utilisés, le type d'encodage utilisé pour vectoriser les tweets et enfin la taille des vecteurs d'encodage des tweets.

Revenons en premier lieu sur la considération des émojis, il y a deux cas possibles, soit supprimer les mots-clés correspondant aux catégories d'émojis, soit affecter un nouveau token à chaque à chaque mot-clé. Lors de la tokenisation que nous avons faite par mot, nous avons réalisé les deux cas pour étudier le poids de ce paramètre pour la classification des émotions.

Le paramètre suivant est la suppression des stopwords classiques en anglais. Les stopwords sont des tokens correspondant aux mots usuels en anglais qui sont très souvent présents mais peu significatifs pour la détections d'émotions dans du texte. En général, leur suppression permet de synthétiser le message et de retirer des informations superflues, les deux cas que nous avons étudiés sont donc : les laisser ou les retirer. Couplé avec les possibilités pour les émojis, nous avons quatre configurations différentes.

Par la suite, nous avons étudié différentes configurations de n-grams. Nous avons fais cela en regroupant en tuples les tokens initiaux correspondant à un seul mot chacun, c'est à dire des 1-gram. Nous avons ainsi traité les 1-gram, 2-grams et 3-grams, puis nous avons réalisé les six configurations possibles : 1-grams seuls, 2-grams seuls, 3-grams seuls, 1-gram et 2-grams, 2-grams et 3-grams, 1-gram et 2-grams et 3-grams. Il va de soi que chaque configuration est couplée aux précédentes obtenues avec les autres paramètres, nous arrivons alors à 24 cas de figures différents.

Avant de vectoriser les tweets, il faut séparer le jeu de données en un jeu d'entraînement et un jeu de test. En effet, la vectorisation tient compte de la fréquence des tokens qu'on lui donne en entrée, or nous souhaitons que cette étape soit totalement indépendante des données de test. La séparation du dataset permet alors que le vectoriseur dépende uniquement des données d'entraînement et pas de test. De plus, nos données ne sont pas réparties équitablement entre chaque classe, nous séparons donc aléatoirement le dataset en conservant les proportions de répartition

des classes.

À cette étape la tokenisation est presque terminée, il ne reste plus qu'à vectoriser chaque tweets. Cependant, la méthode d'encodage des tokens pourrait avoir une influence et pour vérifier cela, nous allons confronter la méthode "One-Hot" qui met en valeur la présence ou l'absence de chaque token (1 si le token est présent dans le tweet 0 sinon) et la méthode "bag-of-word" qui est similaire, mais prend en considération le nombre d'occurrences de chaque token dans le tweet. Cela double le nombre de configuration que nous allons étudier.

Finalement, nous nous intéressons à l'impact de la taille du vecteur d'encodage sur les performances et la précision du modèle. En effet, il est très compliqué de traiter l'ensemble des tokens possibles en raison de la quantité de mémoire nécessaire et du temps de calcul. C'est pourquoi nous conservons uniquement les 1000, 3000, 5000 ou 10000 tokens les plus fréquents rencontrés lors de l'entraînement du vectoriseur. Cette étape se fait évidemment en fonction de la fréquence des tokens dans le jeu d'entraînement et ne tient pas compte des tokens du jeu de test.

## B. ENTRAÎNEMENT ET MODÉLISATION

Notre méthode pour chaque configuration est : de tokeniser les données selon les paramètres de la configuration, de séparer aléatoirement le dataset (20% test et 80% entraînement), de vectoriser les données puis d'entraîner le modèle (LinearSVC de scikitlearn), enfin nous mesurons les performances et la précision. Ce processus est reproduit à 5 reprises pour chaque configuration dans l'objectif de fiabiliser nos résultats et nous moyennons les mesures réalisées pour obtenir une valeur par mesure et par configuration.

Les mesures que nous réalisons sont : la précision, le rappel et la f1-mesure par classe et globale, ainsi que le temps de tokenisation, de vectorisation, d'entraînement du modèle et de prédiction.

En parallèle, nous avons mesuré la précision, le rappel et la f1-mesure par classe et globale de l'implémentation de Bert que nous avons utilisé [4]. Nous avons aussi mesuré le temps de prédiction de ce modèle sur CPU. Il reste cependant important de noter que ce modèle n'a pas été fine-tuné sur nos données, il n'est donc très certainement pas au maximum de ses capacités.

## VI. RÉSULTATS

### A. PRÉCISION DE PRÉDICTION

Dans un premier temps, avant de nous attarder sur les variations de performances techniques, nous avons voulu comparer les résultats de prédiction ; car c'est là l'essence même de notre projet. Parmi toutes les mesures de précision que nous avons calculées (précision, rappel et f1-mesure globale et pour chaque classe d'émotion) nous avons fait le choix de nous attarder sur la f1-mesure. Cette dernière était un bon point moyen de visualiser et d'obtenir un compris entre la précision et le rappel (la précision donnant des informations sur la capacité du modèle à ne pas viser faux et le rappel sur celle à atteindre ça cible ; dans le premier cas on compare les tweets correctement associés à une classe avec le nombre total de tweets associés à cette classe par notre modèle et dans le second on le compare avec le nombre correct total de tweets devaient être associés à cette classe)

$$F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} = 2 \frac{Precision \cdot Recall}{Precision + Recall}$$

FIGURE 1 – Formule utilisée pour le calcul de la valeur f1

#### A.1. ÉTUDE DES ÉMOJIS

Nous avons mené différentes séries de tests afin d'évaluer l'impact de la prise en compte des émojis dans la classification de nos tweets. Dans un cas les tweets comprenant des émojis étaient nettoyés : nous supprimer les émoticônes en laissant le texte intact. Dans un second cas nous avions prévu un dictionnaire regroupant les émojis en différentes catégories (voir annexes). Les émojis étaient donc remplacé par un terme constitué du mot-clé de la catégorie et du suffixe '-smiley' afin de garder l'émotion véhiculée lors de la tokenisation.

À première vue les tests sont relativement similaires et la présence ou non des émojis ne semble pas impacter les performances de prédiction. Cependant, pour approfondir nos résultats, nous avons essayé de récupérer les termes ayant le plus d'impact positif et négatif dans notre classification. Les résultats montrent une certaine influence des émojis, surtout du côté des émotions positives. Pourtant les performances de prédiction vont à l'encontre de ces résultats.

Nous pensons que ces résultats s'expliquent par le fait que nous utilisons les émojis dans le but de compléter nos messages et non comme coeur même du

message. Ainsi la présence ou non d'émoticones aurait pour rôle de conforter la prédiction réalisée grâce au corps du tweet plutôt qu'un rôle décisif. Nous n'excluons pas la possibilité que notre traitement des émojis ait un rôle inhibiteur dans la prédiction. Les émojis étant intégrés dans la norme UTF-8 il pourrait être pertinent d'essayer de les transformer directement en token sans passer par notre phase de catégorisation.

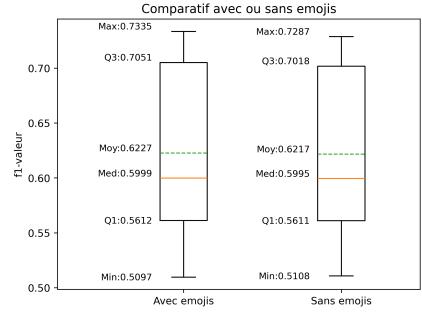


FIGURE 2 – Émojis : f1-valeurs

3.3732	awesome	-2.5795	bad
3.1909	love	-2.3185	terrible
3.0610	[😊, 😊]	-2.3003	suck
3.0051	great	-1.9767	psndown
2.8990	best	-1.8632	blackfriday
2.7937	amaze	-1.8497	poor
2.7281	congratulation	-1.8420	[😢, 😢, 😢, 😢, 😢, 😢, 😢]
2.6516	excellent	-1.8383	middle
2.5276	amazing	-1.7983	garbage
2.3623	perfect	-1.7558	sad
2.3310	[😊, 😊, 😊, 😊]	-1.7509	frustrate
2.3029	nice	-1.7338	facebookdown
2.3004	thanks	-1.6632	properly
2.2814	incredible	-1.6496	wtf
2.2728	apple	-1.6303	whats

FIGURE 3 – 15 mots les plus impactant positivement et négativement

#### A.2. ÉTUDE DES STOPWORDS

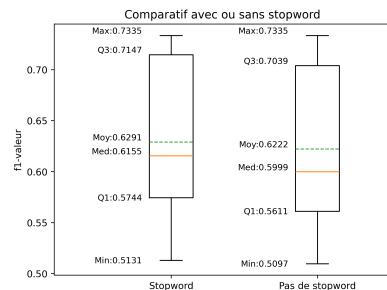


FIGURE 4 – Stopwords : f1-valeurs

Pour la prise en compte, ou non, des stopwords nous n'avons pas eu à faire de pré-traitement particulier. Un paramètre nous permettait de les considérer ou de passer outre une liste de stopwords prédéfinie pour la langue anglaise ([3]). Les résultats montrent une légère amélioration, surtout dans les configurations moyennes, de la prédiction pour les configurations gardant les stopwords. Cela est étonnant puisque ils sont définis comme de l'information superflue. (insérer une analyse)

### A.3. ÉTUDE DU TYPE DE VECTORISATION

Pour la vectorisation nous avons dans un cas (bag-of-word), pris en compte le nombre d'occurrences de chaque token. Et dans un autre (One-Hot), nous avons notifié la présence ou l'absence de chaque token de notre répertoire au sein d'un tweet ; nous perdons aussi l'information de fréquence mais gardons un lien entre les tokens. Les résultats montrent des performances similaires pour les deux types de vectorisation. Nous pensons que cela en dû au contexte de classification ; en effet, les tweets analysés étant des textes courts et traitant de sujets familiers, les vecteurs résultant des deux types d'encodage ne diffèrent pas suffisamment pour observer un mieux.

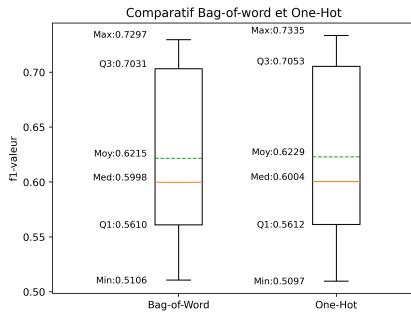


FIGURE 5 – Vectorisation : f1-valeurs

### A.4. ÉTUDE DES NGRAMS

Pour les différents grammages nous avons pris en compte les 1-grams (ce qui équivaut aux tokens seuls) et les associations de tokens par 2 et par 3 (respectivement 2-grams et 3-grams). Nous tenions, tout particulièrement, à également tester les combinaisons de grammages, c'est à dire à la fois 1-grams et 2-grams, 2-grams et 3-grams et aussi 1-2-3 grams ensemble. Non pas que nous pensions qu'une association de différents n-grams serait une révolution dans le monde des tokenizers mais plutôt car cela nous aiderait à comprendre l'influence de ce paramètre.

Il est intéressant de constater que plus on prend en considération des grams élevés, plus la valeur f1 devient mauvaise, pour atteindre ce pic de contre-performance avec les 3-grams seuls. Il peut être pertinent, dans ce cas-ci, de prendre indépendamment les mesures desquelles découle cette f1-valeur pour comprendre l'origine de telles variations (précision et rappel pour le modèle 3 grams : figure 7 et 8).

On constate que le modèle 3-grams présente une précision largement supérieure aux autres modèles mais également un rappel nettement plus faible. Si l'on regarde la précision de différents modèles en fonction des classes d'émotion (figure 9), il ressort que la classe 'neutre' obtient à chaque fois la meilleure pré-

cision. Ce biais, que l'on peut en partie associer à l'écart de proportion entre les classes dans le dataset semble fortement accentué pour le modèle 3-grams. Si l'on évalue la précision par classe pour le meilleur des modèles 3-grams (figure 10) on remarque que seul la précision de la classe neutre est élevée. Les modèles 3-grams ont donc une tendance à tout classifier comme neutre.

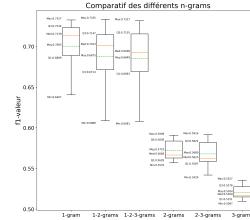


FIGURE 6 – Grammage : f1-valeurs  
FIGURE 7 – Grammage : précisions

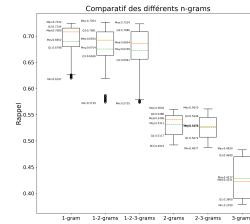


FIGURE 8 – Grammage : rappels

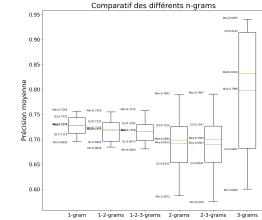


FIGURE 9 – 3-grams : précisions par classe

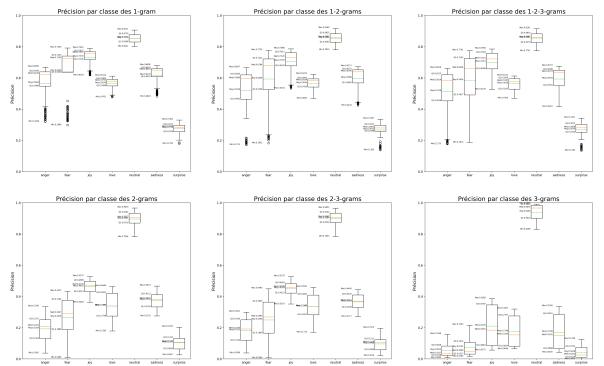


FIGURE 10 – Grammage : précisions par classe

Si l'on reprend notre comparatif des f1-valeurs on constate que les modèles intégrant les tokens seuls (1-grams) sont les plus performants. Cela est vrai également pour le modèle prenant en compte les 1-gram, 2-grams et 3-grams. Cela s'expliquent par le fait que le dernier paramètre que l'on teste est le filtre des tokens avec les N tokens les plus fréquent (les 1-gram sont donc majoritaires). Il apparaît donc que, ici, les tokens seuls sont les plus porteurs d'informations pertinentes et que vouloir associer différents grammages à pour effet d'augmenter la confusion du modèle.

### A.5. ÉTUDE DU NOMBRE DE VECTEURS

Il s'agit ensuite, d'évaluer l'impact du nombre de vecteurs pris en compte lors de la vectorisation sur les performances de notre modèles. Nous avons fait le choix de tester plusieurs cas de figure en limitant la vectorisation aux 1000, 3000, 5000 et 10000 vecteurs les plus pertinents. Cette étude a à la fois pour but d'analyser les conséquences sur la prédiction mais aussi sur les performances techniques de notre modèle ; en effet, un grand nombre de vecteurs a également pour conséquence d'alourdir notre modèle.

Les résultats montrent que prendre en compte un maximum de vecteurs n'est pas forcément synonyme d'efficacité (cela n'est pas sans rappeler le résultat des 3-grams). On constate aussi que 1000 vecteurs est trop peu et que les résultats en pâtissent. Ce sont donc 3000 et 5000 qui présentent les meilleurs résultats de prédiction.

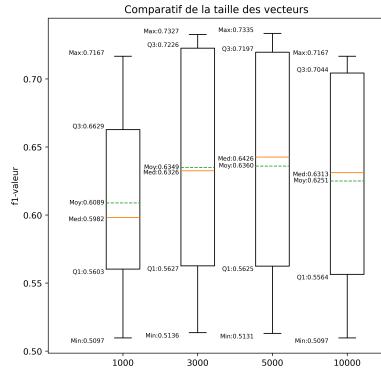


FIGURE 11 – Nombre de vecteurs : f1-valeurs

## B. PERFORMANCE TECHNIQUES

En terme de performances techniques nous avons évalué, pour un même CPU, le temps de tokenisation, de vectorisation, d'entraînement et de prédiction. Le temps de vectorisation étant trop faible et celui de prédiction trop constant pour les relier aux choix de paramètres nous nous sommes concentrés sur le temps de tokenisation et d'entraînement. Notre but était d'étudier l'influence des configurations de paramètres sur ces temps et de les comparer avec les résultats de performances de prédiction.

### B.1. ÉTUDE DES ÉMOJIS

Le temps d'entraînement des modèles est sensiblement le même avec ou sans les émojis. Cependant, on constate que les inclure est en moyenne plus rapide pour la tokenisation. Nous n'expliquons pas ce résultat, cependant il va de paire avec la précision légèrement meilleures en comprenant les émojis.

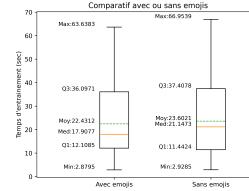


FIGURE 12 – Émojis : Temps d'entraînement

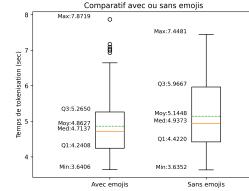


FIGURE 13 – Émojis : Temps de tokenisation

### B.2. ÉTUDE DES STOPWORDS

Un temps de tokenisation plus faible pour les modèles s'étant débarrassés des stopwords est cohérent avec l'élimination prématuée des mots superflus. Cependant on remarque que cela engendre un temps d'entraînement relativement plus long, tandis que nous avions conclus que cela n'impactait presque pas les performances de prédiction dans notre cas.

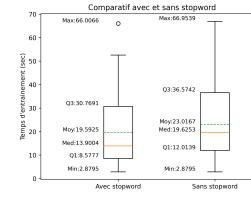


FIGURE 14 – Stopwords : temps d'entraînement

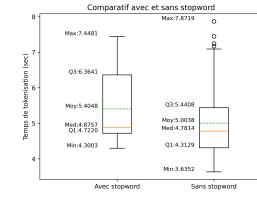


FIGURE 15 – Stopwords : temps de tokenisation

### B.3. ÉTUDE DU TYPE DE VECTORISATION

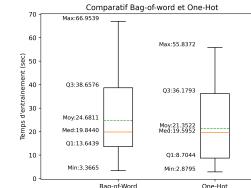


FIGURE 16 – Vectorisation : temps d'entraînement

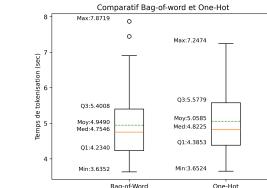


FIGURE 17 – Vectorisation : temps de tokenisation

En moyenne, le type de vectorisation n'a pas de réelle incidence sur les performances techniques. On retrouve là notre constat émis lors de la comparaison des performances de prédiction, le cas précis d'analyse de tweets généraux ne permet pas une distinction concrète entre les 2 types de vectorisation.

### B.4. ÉTUDE DES NGRAMS

Si le temps d'entraînement reste relativement constant en fonction du grammage, on observe un temps de tokenisation plus rapide pour les modèles utilisant des 1-gram. Les 1-gram correspondant aux tokens seuls, ces modèles ne font pas intervenir une

charge de travail supplémentaire lors de cette étape ce qui justifie cette vitesse. Cette vitesse accrue va de paire avec les performances améliorées en terme de prédiction des 1-grams dans l'optique d'une configuration optimale.

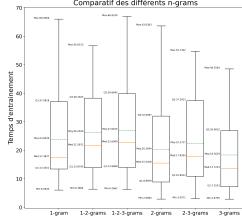


FIGURE 18 –  
Grammage : temps  
d’entraînement

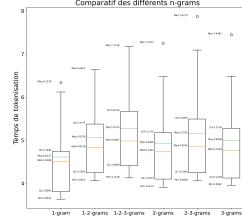


FIGURE 19 –  
Grammage : temps de  
tokenisation

### B.5. ÉTUDE DU NOMBRE DE VECTEURS

Comme on pouvait s'y attendre les modèles prenant en compte 10000 vecteurs sont alourdis et ont un temps de traitement plus long. Cependant les résultats concernant les 1000 vecteurs sont étonnantes, ils semblent très répandus et plutôt élevés. La comparaison entre les modèles 3000 vecteurs et 5000 vecteurs est intéressante. Si l'on avait conclue de l'analyse de performances de prédiction que les deux modèles se valaient, ici on constate clairement que le modèle prenant en considération seulement 3000 vecteurs est plus rapide, à la fois dans la tokenisation que dans l'entraînement.

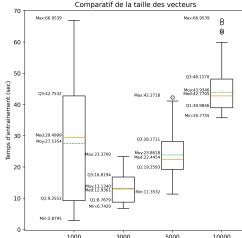


FIGURE 20 – Nombre  
de vecteurs : temps  
d’entraînement

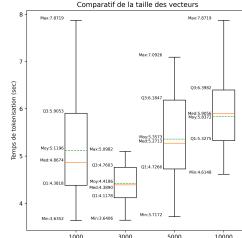


FIGURE 21 – Nombre  
de vecteurs : temps de  
tokenisation

### C. RÉSULTATS ROBERTA

Nous avons utilisé un modèle RoBerta, dérivé du modèle Bert, et fine-tuné pour de la détection d'émotions au sein de tweets en anglais. Les résultats sont, comme l'on pourrait s'y attendre, très bons. Nous avons réduit notre jeu de test pour ne garder que les émotions supportées par le modèle et les résultats indiquent une valeur moyenne d'environ 90% de f1-valeur pour chaque classe. Cependant, les performances techniques ne sont pas comparable.

En effet, cela nous a pris plus de 30h pour effectuer l'analyse complète de notre jeu de données.

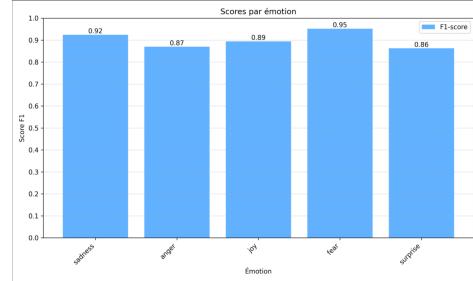


FIGURE 22 – RoBerta : f1-valeurs par classe

## VII. DISCUSSIONS

### A. ANALYSE DES PARAMÈTRES

Grâce à l'ensemble de ces tests il nous possible de discuter l'influence des différents paramètres sur la détection de sentiments dans des tweets. Il est important de prendre en compte le contexte d'expérimentation ; nous cherchons à classifier des tweets, donc des textes courts, dont les sujets sont très généraux et ne requérant aucune connaissance ou compétence particulière. Dans notre cas, on a pu constater que prendre en compte les emojis améliorait légèrement à la fois les prédictions et les performances techniques. Dans un contexte de textes très courts il semblerait que l'étape supplémentaire consistant à filtrer les stopwords n'est pas de réelle valeur ajoutée, au contraire cela ralentirait notre modèle en rallongeant le temps d'entraînement. L'aspect court des tweets a également pour conséquence de se comporter presque identiquement avec les différents types de vectorisation que sont 'bag-of-word' et 'One-Hot'. À propos du grammage, il est clairement apparu que dans notre cas il était plus pertinent d'éviter les grammages élevés et de se tourner vers les 1-gram. Concernant le nombre de vecteurs à prendre en compte, nous avons pu constater que 3000 vecteurs était un bon compromis alliant efficacité et vitesse.

En supposant que nous privilégions la performance de prédiction à celles techniques, la configuration optimale pour de l'analyse d'émotions au sein de tweets serait : avec les emojis / avec les stopwords / vectorisation de type 'Bag-of-Word' / des 1-grams / 3000 vecteurs significatifs

### B. CONFIGURATION OPTIMALE

En moyenne la configuration à avoir obtenu les meilleurs résultats est : avec emojis / avec stopwords / 'One-Hot' / 1-grams / 3000 vecteurs. Elle obtient une moyenne de 73% (pour la mesure de la

f1-valeur). Cette configuration est en tous points similaire à notre meilleure configuration théorique ; exceptée pour le type de vectorisation, bien que son influence soit minime. Cela nous montre que les paramètres testés sont indépendants les uns des autres.

## VIII. CONCLUSION ET TRAVAUX FUTURS

### A. CONCLUSION

En conclusion notre étude nous permet l'accès à de nombreux résultats. En se fiant à la f1-valeur, nous avons des résultats compris entre [0.5097 ; 0.7335]. Bien sûr nos travaux pourrait être complétés avec l'évaluation de plus de paramètres (ou différemment). Cependant, nous nous sommes confronté à une limite liée à l'algorithme de classification. Grâce à cette amélioration du pré-processing notre modèle simple atteint au mieux des performances de 0.73 (f1-valeur) mais est encore loin d'un modèle profond capable de saisir le contexte des tweets (environ 0.90 de f1-valeurs). En revanche, notre modèle et sa configuration optimale ont l'avantage non-négligeable de ne nécessiter, ni un processeur de compétition, ni une nuit de calcul pour assurer une prédiction. Nous voyons là un beau compromis entre vitesse et utilisabilité par le plus grand nombre.

### B. LIMITES ET TRAVAUX FUTURS

Il est important de prendre en compte le contexte restreint (Tweets) des tests. De plus certains paramètres n'ont pas été pris en considération comme le type de tokenisation (word ou subword) ainsi que la présence ou non de stemming et de lemmatisation. Différentes façons de traiter les paramètres sont aussi envisageables comme transformer directement les émojis en ajoutant de nouveaux tokens. Il est également possible de modifier le nettoyage des données.

Nous avons rencontrés deux difficultés dans ce projet. La première étant l'utilisation d'ironie et de second degré, dans le texte mais également dans les émojis, qui est difficile à interpréter. La seconde étant la présence d'un vocabulaire spécifique à la communauté internet et encore plus à la sphère Twitter. Si la première est difficilement outre-passable avec un modèle léger (SVM), la seconde peut être compensée grâce à un dictionnaire spécifique à Twitter afin d'éviter de perdre un maximum d'information ou même de confondre de l'information.

## IX. UTILISATION NÉFASTE OU NOCIVE

L'analyse et la classification des sentiments et des émotions dans les tweets à faible coût peut, aujour-

d'hui, permettre à tous de connaître l'opinion publique sur un sujet donnée. Un individu ou une organisation mal intentionnée pourrait alors choisir un sujet identifié comme sensible pour répandre de fausses informations et amplifier les réaction des utilisateurs.

L'identification des contenus haineux sur une personne pourrait permettre d'assainir une plateforme. Mais un réseau social ou une organisation mal intentionnée pourrait s'en servir pour détruire la réputation ou harceler des personnalités publiques en augmentant la visibilité de ces contenus.

C'est pourquoi, l'étude des émotions dans les tweets et plus particulièrement sur les réseaux sociaux est un puissant outil d'étude des populations qui doit être utilisé avec précautions.

## X. REMERCIEMENTS

Nous souhaitons remercier Sylvie Ratté pour son suivi de notre projet et ses conseils chaque semaine.

## RÉFÉRENCES

- [1] Sanghyun Choo and Wonjoon Kim. A study on the evaluation of tokenizer performance in natural language processing. *Applied Artificial Intelligence*, 37(1) :2175112, 2023. <https://doi.org/10.1080/08839514.2023.2175112>.
- [2] Vandita Grover. Exploiting emojis in sentiment analysis : A survey. *Journal of The Institution of Engineers (India) : Series B*, 103, 2022. <https://doi.org/10.1007/s40031-021-00620-7>.
- [3] Sebastian Leier. Github gist : Regular expressions for tokenization. <https://gist.github.com/sebleier/554280>, 2016. <https://gist.github.com/sebleier/554280>.
- [4] Daniel Loureiro, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and Jose Camacho-collados. {T}ime{LM}s : Diachronic language models from {T}witter. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics : System Demonstrations*, pages 251–260, Dublin, Ireland, may 2022. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2022.acl-demo.25>.
- [5] Vijayarani Mohan. Text mining : Open source tokenization tools : An analysis. *Advanced Computational Intelligence : An International Journal (ACII)*, 3 :37–47, Jan 2016. <https://doi.org/10.5121/acii.2016.3104>.
- [6] Karar Shah. Text sentiment classification, Sep 2023. <https://www.kaggle.com/datasets/kararshah/text-sentiment>.

# L'impact de la tokenisation sur la détection de sentiments

Hugo Laface

*Maîtrise en Technologies de l'Information  
École de Technologie Supérieure  
Montréal, QC, Canada  
hugo.laface.1@ens.etsmtl.ca*

Raphaël Largeau

*Maîtrise en Technologies de l'Information  
École de Technologie Supérieure  
Montréal, QC, Canada  
raphael.largeau.1@ens.etsmtl.ca*

**Résumé**—L'analyse de sentiments à partir de tweets est une tâche complexe attribuée à divers modèles de Natural Language Processing (Naive Bayes, k-Nearest Neighbor, Support Vector Machine, Artificial Neural Networks etc.). Dans le cadre de l'étude de ces modèles nous avons comparé différentes paramétrisations de tokenisation et de vectorisation des données. Notre objectif est de mettre en valeur l'influence du type de tokenisation ainsi que celle du dictionnaire employé dans le but d'en ressortir la préparation de données optimale pour de l'analyse de sentiments de tweets en anglais. Au cours de notre étude, il s'est avéré que la volonté de maximiser et complexifier les paramètres (comme le nombre de vecteurs ou le grammage) pouvait rendre les modèles moins précis. À l'inverse, pour le grammage, les tokens seuls (1-gram) ont une meilleure précision. La complexité de la langue et en particulier celle du “langage Twitter” est également un frein à la recherche de performances optimales. Nous avons ainsi pu trouver une configuration des paramètres pour maximiser les performances avec notre modèle léger (machine à vecteurs de support). Finalement, notre modèle, couplé à un pré-processing idéal, reste moins précis dans la détection de sentiments que ne l'est un modèle profond spécialement entraîné pour cette tâche (comme BERT), bien qu'il soit beaucoup plus rapide à entraîner et pour réaliser des prédictions. Nos performances finales sont tout de même bonnes et notre modèle offre un fervent compromis alliant précision et rapidité de prédiction.

**Mots-clés**—Tweets, tokenisation, détection de sentiments, NLP, apprentissage machine

## I. INTRODUCTION

L'analyse des sentiments dans les tweets a de plus en plus d'applications pour un grand nombre d'organisations et d'entreprises. Cela permet, par exemple, d'étudier la satisfaction d'un client ou d'évaluer l'opinion publique sur un sujet. Elle peut même permettre de prévoir l'évolution d'une pandémie en observant les émotions majoritaires sur un sujet précis selon les différentes régions du monde. Cependant, avec les technologies actuelles, une étude précise des sentiments sur de nombreux tweets demande une importante puissance de calcul.

C'est pourquoi, nous souhaitons étudier l'impact de différents tokenizers sur la performance de modèles d'analyse textuelle de sentiments. Notre travail

s'est axé sur l'étude de différents paramètres de préparation et de tokenisation des tweets. Ensuite, pour chaque ensemble de ces paramètres, nous avons entraîné un modèle simple (machine à vecteurs de support). Puis, nous avons comparé ces modèles entre eux selon les ensembles de paramètres et avec un modèle Bert pré-entraîné pour la détection d'émotions au sein de tweets. Finalement, nous répondrons à la question : quel impact le prétraitement et plus particulièrement la tokenization ont ils sur la détection de sentiments dans des données textuelles ?

## II. REVUE DE LITTÉRATURE

La détection de sentiments est un sujet de recherche très répandu et qui témoigne d'un intérêt grandissant depuis la numérisation de la société. Divers travaux sont donc menés dans l'optique d'améliorer cette détection. Un intérêt tout particulier est porté à la phase de pré-processing car jugée moins gourmande en ressource que le processing lui-même. La plupart des recherches menées dans ce cadre visent à comparer des tokenizers existant afin d'en déduire le plus performant. Peu d'entre elles s'intéressent concrètement au fonctionnement profond et à l'impact des (hyper)paramètres sur la tokenisation. On note par exemple les travaux de Sanghyun Choo et Wonjoon Kim dans lesquels ils [1] comparent deux tokenizers (Mecan-ko et SentencePiece) pour de l'analyse de sentiments contenus dans des avis d'achats coréens. Leur approche est différente de la nôtre dans le sens où il recherche le plus performant en association avec plus algorithme de classification (Naive Bayes, k-Nearest Neighbor, Support Vector Machine, Artificial Neural Networks, et Long Short-Term Memory Recurrent Neural Networks). De plus, ils utilisent des tokenizers complets et disposant d'algorithmes et de composants particuliers, couplé à l'usage de 5 modèles différents leurs résultats sont orientés vers l'ultra-performance plutôt qu'un compromis performance-légereté. Dans [5], différents outils de tokenisation

sont comparés dont NLTK que nous utilisons dans notre recherche pour en analyser les performances. Un travail notable mais, une fois de plus, qui ne s'attarde pas sur le fonctionnement profond des outils. Il existe toute fois des travaux menés sur certains paramètres spécifiques, comme par le Dr. Vijayarani Mohan dans [2] qui mène une étude complète sur les émojis au sein de la tokenization. Travail selon lequel les émojis améliore cette détection de sentiments et pourrait avoir un impact sur la compréhension d'émotions plus complexes comme le sarcasme, l'ironie ou l'humour noir.

### III. OBJECTIFS

Au cours de ce projet, nous nous sommes fixé l'objectif d'étudier un maximum de paramètres de préparation et de tokenisation des tweets, puis d'en déduire leurs influences et leurs importances dans la détection de sentiments. Nous avons ensuite divisé cet objectif principal en trois étapes. La première étape consiste à pouvoir appliquer différents paramètres de pré-traitement et de tokenisation sur nos données. Les paramètres que nous souhaitons étudier sont notamment : les n-grams, les stopwords, la présence ou non d'émojis, la méthode de vectorisation des tokens et la taille du vecteur d'encodage. La deuxième étape vise à entraîner un modèle (machine à vecteur de support) avec différentes configuration de paramètres sur nos données afin d'en ressortir des résultats de précision et de performance à comparer. Enfin, la troisième étape concerne l'étude des prédictions d'un modèle plus puissance d'apprentissage profond pré-entraîné (Bert) sur nos données en comparaison des résultats obtenus à l'étape précédente.

### IV. MATÉRIEL

#### A. DONNÉES

Nous avons choisi nos données parmi les datasets de tweets disponible sur Kaggle. Puisque notre projet consiste à pré-traiter au mieux des tweets, nous avions besoin de tweets dans leur état le plus brut. Par ailleurs, nous voulions que ces tweets soient tous annotés avec des émotions plus complexes que "positif" ou "négatif". Notre dernière exigence était d'avoir un grand nombre de tweets pour pouvoir entraîner de manière fiable un modèle.

À partir de ces contraintes, nous avons choisi le dataset [6]. Il est composé de 69298 tweets complètement bruts sur divers sujets, avec parfois des liens, des noms d'utilisateurs et des caractères spéciaux comme des émojis. Ces tweets sont tous rédigés en anglais, ce qui simplifie notre travail. De plus, chaque tweet

est annoté par une émotion parmi 7 différentes. Les émotions possibles avec leur nombre d'instances respectif dans le dataset sont : "neutral" (24659), "sadness" (13029), "Joy" (10076), "Love" (7402), "Anger" (6567), "Surprise" (3933), "Fear" (3632). Ce dataset répond donc bien à nos exigences. Cependant, la proportion de représentation des classes n'est pas équitable, ce qui nous amènera à faire quelques ajustements dans la préparation des données.

#### B. OUTILS

Pour réaliser notre projet, nous avons eu recours à python en raison de la grande variété de bibliothèques que ce langage propose, notamment dans le domaine du traitement des données et de l'apprentissage machine. De plus ce langage nous permet d'utiliser à la fois notre modèle léger (machine à vecteur de support) et le modèle Bert, dans des conditions similaires, ce qui permet de mieux les comparer.

Toute notre phase de préparation et de tokenisation utilise les bibliothèques nltk et transformers. En effet, ces bibliothèques fournissent toutes les fonctions nécessaires au nettoyage et à la tokenisation de tweets selon tous les paramètres que nous visons d'étudier.

Pour notre modèle, nous avons utilisé une machine à vecteur de support fournie par scikitlearn (LinearSVC), cela correspond parfaitement à nos besoins puisque cette bibliothèque est optimisée et permet d'entraîner rapidement de nombreux modèles sur un CPU.

L'implémentation de Bert que nous avons utilisé, quant à elle, a été implémentée avec pytorch et provient de [4]. Nous avons utilisé ce modèle sur CPU pour que les performances soient comparables à celles du modèle précédent.

Finalement, pour tout l'affichage de nos résultats, nous avons fait le choix d'utiliser matplotlib pour l'ensemble des possibilités qu'elle offre, ce qui nous permet de mieux mettre en valeur nos résultats.

### V. MÉTHODES

#### A. PRÉ-TRAITEMENT ET TOKENISATION

Comme nous l'avons précisé plus tôt, nos données sont des tweets bruts sur divers sujets. Avant la tokenisation, il convient donc de les nettoyer. Pour faire cela, nous avons d'abord passé tous les tweets en minuscules. Ensuite, nous avons retiré les liens et les noms d'utilisateurs des tweets en partant de l'hypothèse qu'ils ne portent pas de signification facilement exploitable. Pour simplifier les tweets, nous avons retiré tous les numéros et caractères spéciaux, sauf les

émojis que nous avons seulement regroupés par catégories en les remplaçant par des mot-clés et sur lesquels nous reviendrons plus tard. Nous avons aussi retiré la ponctuation pour simplifier le problème, mais avec plus de temps, nous aurions voulu rajouter des tokens particulier pour certaines formes de ponctuation comme les exclamations et les interrogations.

Une fois la première étape de nettoyage préliminaire effectuée sur tous les tweets, nous avons commencé à appliquer différents paramètres à leur prétraitement et à leur tokenisation. Ces paramètres sont : la prise en compte ou non des émojis, la suppression ou non des stopwords dans les tweets, les n-grams utilisés, le type d'encodage utilisé pour vectoriser les tweets et enfin la taille des vecteurs d'encodage des tweets.

Revenons en premier lieu sur la considération des émojis, il y a deux cas possibles, soit supprimer les mots-clés correspondant aux catégories d'émojis, soit affecter un nouveau token à chaque à chaque mot-clé. Lors de la tokenisation que nous avons faite par mot, nous avons réalisé les deux cas pour étudier le poids de ce paramètre pour la classification des émotions.

Le paramètre suivant est la suppression des stopwords classiques en anglais. Les stopwords sont des tokens correspondant aux mots usuels en anglais qui sont très souvent présents mais peu significatifs pour la détections d'émotions dans du texte. En général, leur suppression permet de synthétiser le message et de retirer des informations superflues, les deux cas que nous avons étudiés sont donc : les laisser ou les retirer. Couplé avec les possibilités pour les émojis, nous avons quatre configurations différentes.

Par la suite, nous avons étudié différentes configurations de n-grams. Nous avons fait cela en regroupant en tuples les tokens initiaux correspondant à un seul mot chacun, c'est à dire des 1-gram. Nous avons ainsi traité les 1-gram, 2-grams et 3-grams, puis nous avons réalisé les six configurations possibles : 1-grams seuls, 2-grams seuls, 3-grams seuls, 1-gram et 2-grams, 2-grams et 3-grams, 1-gram et 2-grams et 3-grams. Il va de soi que chaque configuration est couplée aux précédentes obtenues avec les autres paramètres, nous arrivons alors à 24 cas de figures différents.

Avant de vectoriser les tweets, il faut séparer le jeu de données en un jeu d'entraînement et un jeu de test. En effet, la vectorisation tient compte de la fréquence des tokens qu'on lui donne en entrée, or nous souhaitons que cette étape soit totalement indépendante des données de test. La séparation du dataset permet alors que le vectoriseur dépende uniquement des données d'entraînement et pas de test. De plus,

nos données ne sont pas réparties équitablement entre chaque classe, nous séparons donc aléatoirement le dataset en conservant les proportions de répartition des classes.

À cette étape la tokenisation est presque terminée, il ne reste plus qu'à vectoriser chaque tweets. Cependant, la méthode d'encodage des tokens pourrait avoir une influence et pour vérifier cela, nous allons confronter la méthode "One-Hot" qui met en valeur la présence ou l'absence de chaque token (1 si le token est présent dans le tweet 0 sinon) et la méthode "bag-of-word" qui est similaire, mais prend en considération le nombre d'occurrences de chaque token dans le tweet. Cela double le nombre de configuration que nous allons étudier.

Finalement, nous nous intéressons à l'impact de la taille du vecteur d'encodage sur les performances et la précision du modèle. En effet, il est très compliqué de traiter l'ensemble des tokens possibles en raison de la quantité de mémoire nécessaire et du temps de calcul. C'est pourquoi nous conservons uniquement les 1000, 3000, 5000 ou 10000 tokens les plus fréquents rencontrés lors de l'entraînement du vectoriseur. Cette étape se fait évidemment en fonction de la fréquence des tokens dans le jeu d'entraînement et ne tient pas compte des tokens du jeu de test.

## B. ENTRAÎNEMENT ET MODÉLISATION

Notre méthode pour chaque configuration est : de tokeniser les données selon les paramètres de la configuration, de séparer aléatoirement le dataset en conservant les proportions de répartition des classes (20% test et 80% entraînement), de vectoriser les données puis d'entraîner le modèle (LinearSVC de scikitlearn), enfin nous mesurons les performances et la précision. Ce processus est reproduit à 5 reprises pour chaque configuration dans l'objectif de fiabiliser nos résultats et nous moyennons les mesures réalisées pour obtenir une valeur par mesure et par configuration.

Les mesures que nous réalisons sont : la précision, le rappel et la f1-mesure par classe et globale, ainsi que le temps de tokenisation, de vectorisation, d'entraînement du modèle et de prédiction.

En parallèle, nous avons mesuré la précision, le rappel et la f1-mesure par classe et globale de l'implémentation de Bert que nous avons utilisé [4]. Nous avons aussi mesuré le temps de prédiction de ce modèle sur CPU. Il reste cependant important de noter que ce modèle n'a pas été fine-tuné sur nos données, il n'est donc très certainement pas au maximum de ses capacités.

## VI. RÉSULTATS

### A. PRÉCISION DE PRÉDICTION

Dans un premier temps, avant de nous attarder sur les variations de performances techniques, nous avons voulu comparer les résultats de prédiction car c'est là l'essence même de notre projet. Parmi toutes les mesures de précision que nous avons calculées (précision, rappel et f1-mesure globale et pour chaque classe d'émotion) nous avons fait le choix de nous attarder sur la f1-mesure. Cette dernière était un bon point moyen de visualiser et d'obtenir un compris entre la précision et le rappel (la précision donnant des informations sur la capacité du modèle à ne pas viser faux et le rappel sur celle à atteindre ça cible ; dans le premier cas on compare les tweets correctement associés à une classe avec le nombre total de tweets associés à cette classe par notre modèle et dans le second on le compare avec le nombre correct total de tweets devaient être associés à cette classe)

$$F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} = 2 \frac{Precision \cdot Recall}{Precision + Recall}$$

FIGURE 1 – Formule utilisée pour le calcul de la valeur f1

### A.1. ÉTUDE DES ÉMOJIS

Nous avons mené différentes séries de tests afin d'évaluer l'impact de la prise en compte des émojis dans la classification de nos tweets. Dans un cas les tweets comprenant des émojis étaient nettoyés : nous supprimions les émoticones en laissant le texte intact. Dans un second cas nous avions prévu un dictionnaire regroupant les émojis en différentes catégories (voir annexes). Les émojis étaient donc remplacé par un terme constitué du mot-clef de la catégorie et du suffixe "-smiley" afin de garder l'émotion véhiculée lors de la tokenisation.

À première vue les tests sont relativement similaires et la présence ou non des émojis ne semble pas impacter les performances de prédiction. Cependant, pour approfondir nos résultats, nous avons essayé de récupérer les termes ayant le plus d'impact positif et négatif dans notre classification. Les résultats montrent une certaine influence des émojis, surtout du côté des émotions positives. Pourtant les performances de prédiction vont à l'encontre de ces résultats.

Nous pensons que ces résultats s'expliquent par le fait que les émojis sont utilisés dans le but de compléter nless messages et non comme coeur même du

message. Ainsi, la présence ou non d'émoticones aurait pour rôle de conforter la prédiction réalisée grâce au corps du tweet plutôt qu'un rôle décisif. Nous n'excluons pas la possibilité que notre traitement des émojis ait un rôle inhibiteur dans la prédiction. Les émojis étant intégrés dans la norme UTF-8 il pourrait aussi être pertinent d'essayer de les transformer directement en token sans passer par notre phase de catégorisation.

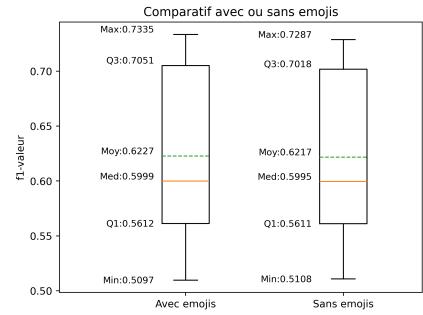


FIGURE 2 – Émojis : f1-valeurs

3.3732	awesome	-2.5795	bad
3.1909	love	-2.3185	suck
3.0610	[😊, 😘]	-2.3003	psdown
3.0051	great	-1.9767	blackfriday
2.8990	best	-1.8632	poor
2.7937	amaze	-1.8497	middle
2.7281	congratulation	-1.8420 [😢, 😔, 😕, 😖, 😔, 😔, 😔, 😔]	garbage
2.6516	excellent	-1.8383	sad
2.5276	amazing	-1.7983	facebookdown
2.3623	perfect	-1.7558	properly
2.3310	[😊, 😘, 😃, 😄]	-1.7509	wtf
2.3029	nice	-1.7338	whats
2.3004	thanks	-1.6632	
2.2814	incredible	-1.6496	
2.2728	!	-1.6303	

FIGURE 3 – 15 mots les plus impactant positivement et négativement

### A.2. ÉTUDE DES STOPWORDS

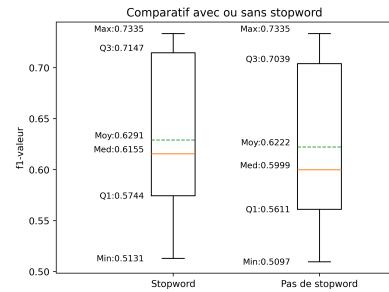


FIGURE 4 – Stopwords : f1-valeurs

Pour la prise en compte, ou non, des stopwords nous n'avons pas eu à faire de pré-traitement particulier. Un paramètre nous permettait de les considérer ou de passer outre une liste de stopwords prédéfinie pour la langue anglaise ([3]). Les résultats montrent une légère amélioration, surtout dans les configurations moyennes, de la prédiction pour les configurations gardant les stopwords. Cela est étonnant puisqu'ils sont définis comme de l'information superflue.

### A.3. ÉTUDE DU TYPE DE VECTORISATION

Pour la vectorisation nous avons dans un cas (bag-of-word), pris en compte le nombre d'occurrences de chaque token. Et dans un autre (One-Hot), nous avons notifié la présence ou l'absence de chaque token de notre répertoire au sein d'un tweet. Nous perdons aussi l'information de fréquence mais gardons un lien entre les tokens. Les résultats montrent des performances similaires pour les deux type de vectorisation. Nous pensons que cela en dû au contexte de classification. En effet, les tweets analysés étant des textes courts et traitant de sujets familiers, les vecteurs résultant des deux types d'encodage ne diffèrent pas suffisamment pour observer un mieux.

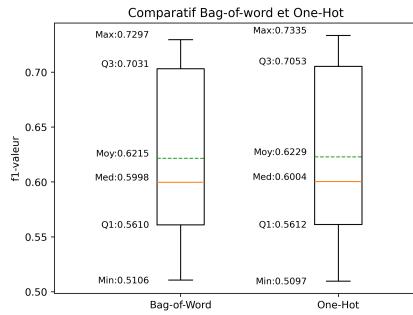


FIGURE 5 – Vectorisation : f1-valeurs

### A.4. ÉTUDE DES NGRAMS

Pour les différents grammages nous avons pris en compte les 1-grams (ce qui équivaut aux tokens seuls) et les associations de tokens par 2 et par 3 (respectivement 2-grams et 3-grams). Nous tenions, tout particulièrement, à également tester les combinaisons de grammages, c'est à dire à la fois 1-grams et 2-grams, 2-grams et 3-grams et aussi 1-2-3 grams ensemble. Non pas que nous pensions qu'une association de différents n-grams serait une révolution dans le monde des tokenizers mais plutôt car cela nous aiderait à comprendre l'influence de ce paramètre.

Il est intéressant de constater que plus on prend en considération des grams élevés, plus la valeur f1 devient mauvaise, pour atteindre ce pic de contre-performance avec les 3-grams seuls. Il peut être pertinent, dans ce cas-ci, de prendre indépendamment les mesures desquelles découle cette f1-valeur pour comprendre l'origine de telles variations (précision et rappel pour le modèle 3 grams : figure 7 et 8).

On constate que le modèle 3-grams présente une précision largement supérieure aux autres modèles mais également un rappel nettement plus faible. Si l'on regarde la précision de différents modèles en fonction des classes d'émotion (figure 9), il ressort que la classe 'neutre' obtient à chaque fois la meilleure pré-

cision. Ce biais, que l'on peut en partie associer à l'écart de proportion entre les classes dans le dataset semble fortement accentué pour le modèle 3-grams. Si l'on évalue la précision par classe pour le meilleur des modèles 3-grams (figure 10) on remarque que seul la précision de la classe neutre est élevée. Les modèles 3-grams ont donc une tendance à tout classifier comme neutre.

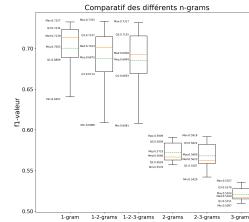


FIGURE 6 – Grammage : f1-valeurs  
FIGURE 7 – Grammage : précisions

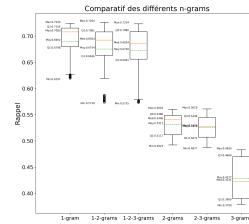


FIGURE 8 – Grammage : rappels

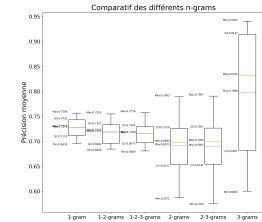


FIGURE 9 – 3-grams : précisions par classe

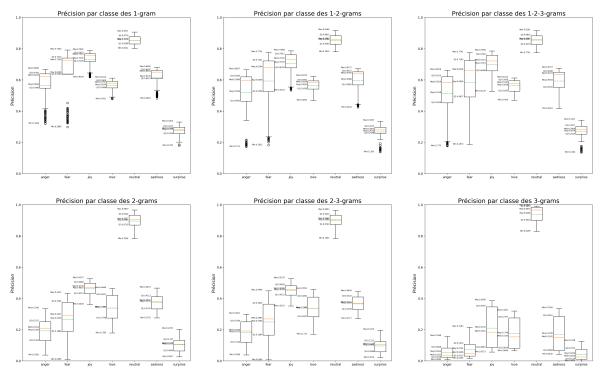


FIGURE 10 – Grammage : précisions par classe

Si l'on reprend notre comparatif des f1-valeurs on constate que les modèles intégrant les tokens seuls (1-grams) sont les plus performants. Cela est vrai également pour le modèle prenant en compte les 1-gram, 2-grams et 3-grams. Cela s'expliquent par le fait que le dernier paramètre que l'on teste est le filtre des tokens avec les N tokens les plus fréquent (les 1-gram sont donc majoritaires). Il apparaît donc que, ici, les tokens seuls sont les plus porteurs d'informations pertinentes et que vouloir associer différents grammages à pour effet d'augmenter la confusion du modèle.

### A.5. ÉTUDE DU NOMBRE DE VECTEURS

Il s'agit ensuite, d'évaluer l'impact du nombre de vecteurs pris en compte lors de la vectorisation sur les performances de notre modèles. Nous avons fait le choix de tester plusieurs cas de figure en limitant la vectorisation aux 1000, 3000, 5000 et 10000 vecteurs les plus pertinents. Cette étude a à la fois pour but d'analyser les conséquences sur la prédiction mais aussi sur les performances techniques de notre modèle. En effet, un grand nombre de vecteurs a également pour conséquence d'alourdir notre modèle.

Les résultats montrent que prendre en compte un maximum de vecteurs n'est pas forcément synonyme d'efficacité (cela n'est pas sans rappeler le résultat des 3-grams). On constate aussi que 1000 vecteurs est trop peu et que les résultats en pâtissent. Ce sont donc 3000 et 5000 qui présentent les meilleurs résultats de prédiction.

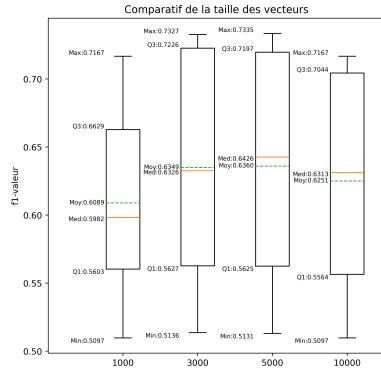


FIGURE 11 – Nombre de vecteurs : f1-valeurs

## B. PERFORMANCE TECHNIQUES

En terme de performances techniques nous avons évalué, pour un même CPU, le temps de tokenisation, de vectorisation, d'entraînement et de prédiction. Le temps de vectorisation étant trop faible et celui de prédiction trop constant pour les relier aux choix de paramètres nous nous sommes concentrés sur le temps de tokenisation et d'entraînement. Notre but était d'étudier l'influence des configurations de paramètres sur ces temps et de les comparer avec les résultats de performances de prédiction.

### B.1. ÉTUDE DES ÉMOJIS

Le temps d'entraînement des modèles est sensiblement le même avec ou sans les émojis. Cependant, on constate que les inclure est en moyenne plus rapide pour la tokenisation. Nous n'expliquons pas ce résultat, cependant il va de paire avec la précision légèrement meilleures en comprenant les émojis.

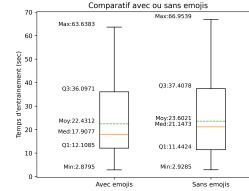


FIGURE 12 – Émojis : temps d'entraînement

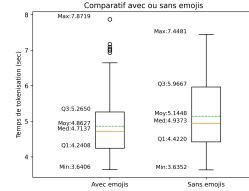


FIGURE 13 – Émojis : temps de tokenisation

### B.2. ÉTUDE DES STOPWORDS

Un temps de tokenisation plus faible pour les modèles s'étant débarrassés des stopwords est cohérent avec l'élimination prématûre des mots superflus. Cependant on remarque que cela engendre un temps d'entraînement relativement plus long, tandis que nous avions conclus que cela n'impactait presque pas les performances de prédiction dans notre cas.

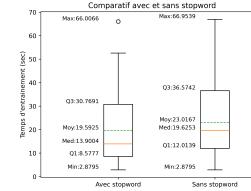


FIGURE 14 – Stopwords : temps d'entraînement

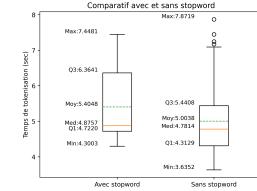


FIGURE 15 – Stopwords : temps de tokenisation

### B.3. ÉTUDE DU TYPE DE VECTORISATION

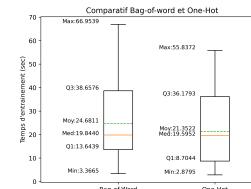


FIGURE 16 – Vectorisation : temps d'entraînement

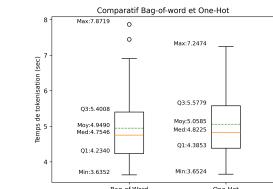


FIGURE 17 – Vectorisation : temps de tokenisation

En moyenne, le type de vectorisation n'a pas de réelle incidence sur les performances techniques. On retrouve là notre constat émis lors de la comparaison des performances de prédiction, le cas précis d'analyse de tweets généraux ne permet pas une distinction concrète entre les 2 types de vectorisation.

### B.4. ÉTUDE DES NGRAMS

Si le temps d'entraînement reste relativement constant en fonction du grammage, on observe un temps de tokenisation plus rapide pour les modèles utilisant des 1-gram. Les 1-gram correspondant aux tokens seuls, ces modèles ne font pas intervenir une

charge de travail supplémentaire lors de cette étape ce qui justifie cette vitesse. Cette vitesse accrue va de paire avec les performances améliorées en terme de prédiction des 1-grams dans l'optique d'une configuration optimale.

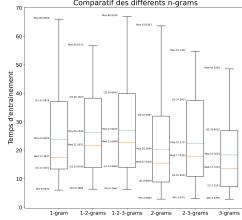


FIGURE 18 –  
Grammage : temps  
d’entraînement

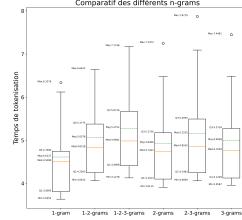


FIGURE 19 –  
Grammage : temps de  
tokenisation

### B.5. ÉTUDE DU NOMBRE DE VECTEURS

Comme on pouvait s'y attendre les modèles prenant en compte 10000 vecteurs sont alourdis et ont un temps de traitement plus long. Cependant les résultats concernant les 1000 vecteurs sont étonnantes, ils semblent très répandus et plutôt élevés. La comparaison entre les modèles 3000 vecteurs et 5000 vecteurs est intéressante. Si l'on avait conclue de l'analyse de performances de prédiction que les deux modèles se valaient, ici on constate clairement que le modèle prenant en considération seulement 3000 vecteurs est plus rapide, à la fois dans la tokenisation que dans l'entraînement.

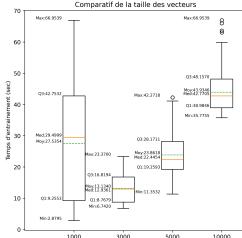


FIGURE 20 – Nombre de vecteurs : temps d’entraînement

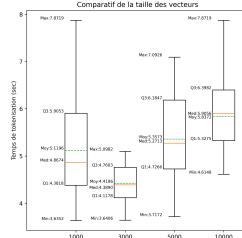


FIGURE 21 – Nombre de vecteurs : temps de tokenisation

### C. RÉSULTATS ROBERTA

Nous avons utilisé un modèle RoBerta, dérivé du modèle Bert, et fine-tuné pour de la détection d'émotions au sein de tweets en anglais. Les résultats sont, comme l'on pourrait s'y attendre, très bons. Nous avons réduit notre jeu de test pour ne garder que les émotions supportées par le modèle et les résultats indiquent une valeur moyenne d'environ 90% de f1-valeur pour chaque classe. Cependant, les performances techniques ne sont pas comparable.

En effet, cela nous a pris plus de 30h pour effectuer l'analyse complète de notre jeu de données.

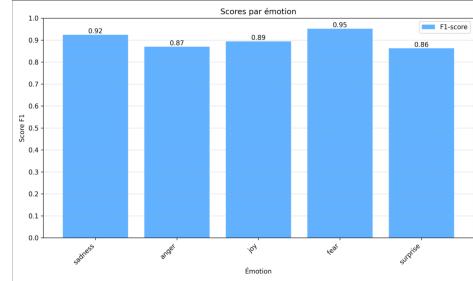


FIGURE 22 – RoBerta : f1-valeurs par classe

## VII. DISCUSSIONS

### A. ANALYSE DES PARAMÈTRES

Grâce à l'ensemble de ces tests il nous possible de discuter l'influence des différents paramètres sur la détection de sentiments dans des tweets. Il est important de prendre en compte le contexte d'expérimentation ; nous cherchons à classifier des tweets, donc des textes courts, dont les sujets sont très généraux et ne requérant aucune connaissance ou compétence particulière. Dans notre cas, on a pu constater que prendre en compte les emojis améliorait légèrement à la fois les prédictions et les performances techniques. Dans un contexte de textes très courts il semblerait que l'étape supplémentaire consistant à filtrer les stopwords n'est pas de réelle valeur ajoutée, au contraire cela ralentirait notre modèle en rallongeant le temps d'entraînement. L'aspect court des tweets a également pour conséquence de se comporter presque identiquement avec les différents types de vectorisation que sont 'bag-of-word' et 'One-Hot'. À propos du grammage, il est clairement apparu que dans notre cas il était plus pertinent d'éviter les grammages élevés et de se tourner vers les 1-gram. Concernant le nombre de vecteurs à prendre en compte, nous avons pu constater que 3000 vecteurs était un bon compromis alliant efficacité et vitesse.

En supposant que nous privilégions la performance de prédiction à celles techniques, la configuration optimale pour de l'analyse d'émotions au sein de tweets serait : avec les emojis / avec les stopwords / vectorisation de type 'Bag-of-Word' / des 1-grams / 3000 vecteurs significatifs

### B. CONFIGURATION OPTIMALE

En moyenne la configuration à avoir obtenu les meilleurs résultats est : avec emojis / avec stopwords / 'One-Hot' / 1-grams / 3000 vecteurs. Elle obtient une moyenne de 73% (pour la mesure de la

f1-valeur). Cette configuration est en tous points similaire à notre meilleure configuration théorique ; exceptée pour le type de vectorisation, bien que son influence soit minime. Cela nous montre que les paramètres testés sont indépendants les uns des autres.

## VIII. CONCLUSION ET TRAVAUX FUTURS

### A. CONCLUSION

En conclusion notre étude nous permet l'accès à de nombreux résultats. En se fiant à la f1-valeur, nous avons des résultats compris entre [0.5097 ; 0.7335]. Bien sûr nos travaux pourrait être complétés avec l'évaluation de plus de paramètres (ou différemment). Cependant, nous nous sommes confronté à une limite liée à l'algorithme de classification. Grâce à cette amélioration du pré-processing notre modèle simple atteint au mieux des performances de 0.73 (f1-valeur) mais est encore loin d'un modèle profond capable de saisir le contexte des tweets (environ 0.90 de f1-valeurs). En revanche, notre modèle et sa configuration optimale ont l'avantage non-négligeable de ne nécessiter, ni un processeur de compétition, ni une nuit de calcul pour assurer une prédiction. Nous voyons là un beau compromis entre vitesse et utilisabilité par le plus grand nombre.

### B. LIMITES ET TRAVAUX FUTURS

Il est important de prendre en compte le contexte restreint (Tweets) des tests. De plus certains paramètres n'ont pas été pris en considération comme le type de tokenisation (word ou subword) ainsi que la présence ou non de stemming et de lemmatisation. Différentes façons de traiter les paramètres sont aussi envisageables comme transformer directement les émojis en ajoutant de nouveaux tokens. Il est également possible de modifier le nettoyage des données.

Nous avons rencontrés deux difficultés dans ce projet. La première étant l'utilisation d'ironie et de second degré, dans le texte mais également dans les émojis, qui est difficile à interpréter. La seconde étant la présence d'un vocabulaire spécifique à la communauté internet et encore plus à la sphère Twitter. Si la première est difficilement outre-passable avec un modèle léger (SVM), la seconde peut être compensée grâce à un dictionnaire spécifique à Twitter afin d'éviter de perdre un maximum d'information ou même de confondre de l'information.

## IX. UTILISATION NÉFASTE OU NOCIVE

L'analyse et la classification des sentiments et des émotions dans les tweets à faible coût peut, aujour-

d'hui, permettre à tous de connaître l'opinion publique sur un sujet donnée. Un individu ou une organisation mal intentionnée pourrait alors choisir un sujet identifié comme sensible pour répandre de fausses informations et amplifier les réaction des utilisateurs.

L'identification des contenus haineux sur une personne pourrait permettre d'assainir une plateforme. Mais un réseau social ou une organisation mal intentionnée pourrait s'en servir pour détruire la réputation ou harceler des personnalités publiques en augmentant la visibilité de ces contenus.

C'est pourquoi, l'étude des émotions dans les tweets et plus particulièrement sur les réseaux sociaux est un puissant outil d'étude des populations qui doit être utilisé avec précautions.

## X. REMERCIEMENTS

Nous souhaitons remercier Sylvie Ratté pour son suivi de notre projet et ses conseils chaque semaine.

## RÉFÉRENCES

- [1] Sanghyun Choo and Wonjoon Kim. A study on the evaluation of tokenizer performance in natural language processing. *Applied Artificial Intelligence*, 37(1) :2175112, 2023. <https://doi.org/10.1080/08839514.2023.2175112>.
- [2] Vandita Grover. Exploiting emojis in sentiment analysis : A survey. *Journal of The Institution of Engineers (India) : Series B*, 103, 2022. <https://doi.org/10.1007/s40031-021-00620-7>.
- [3] Sebastian Leier. Github gist : Regular expressions for tokenization. <https://gist.github.com/sebleier/554280>, 2016. <https://gist.github.com/sebleier/554280>.
- [4] Daniel Loureiro, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and Jose Camacho-collados. {T}ime{LM}s : Diachronic language models from {T}witter. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics : System Demonstrations*, pages 251–260, Dublin, Ireland, may 2022. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2022.acl-demo.25>.
- [5] Vijayarani Mohan. Text mining : Open source tokenization tools : An analysis. *Advanced Computational Intelligence : An International Journal (ACII)*, 3 :37–47, Jan 2016. <https://doi.org/10.5121/acii.2016.3104>.
- [6] Karar Shah. Text sentiment classification, Sep 2023. <https://www.kaggle.com/datasets/kararshah/text-sentiment>.