Université Cadi Ayyad
École Nationale des Sciences Appliquées
Marrakech

Rapport du Projet de Fin de semestre

Réalisation d'une application d'analyse des sentiments des tweets en temps réel

Réalisé par :
Ayoub LABIAD
Zakaria MESTAOUI

Encadré par :
Pr. Mustapha AMEUR

Jury :

Pr. Mustapha AMEUR
Pr. Abderrazzak NEJEOUI

2019/2020

Table des matières

Résumé

Trouver un framework pour prédire le cours des actions est une priorité sur le marché financier. Durant ce stage, nous proposons un framework utilisant des réseaux antagonistes génératifs (GANs) pour prévoir les cours des actions. Notre framework utilise la mémoire à court terme long (LSTM) comme générateur et le réseau neuronal convolutif (CNN) comme discriminateur. Le générateur est formé sur des données de bruit aléatoires au lieu des données réelles, pour essayer de prévoir les prix futurs de la bourse. Nos résultats montrent la possibilité d'utiliser des GANs pour prédire les cours des actions et les performances de notre modèle par rapport à d'autres approches.

Remerciements

Tout d'abord, nous tenons à remercier notre encadrant Pr. Mustapha AMEUR, qui nous a accompagnés tout au long de cette période du projet, par son suivi régulier, ses conseils pertinents quant à l'organisation et l'avancement de notre étude et sa disponibilité pour répondre à nos questions.

Nous tenons à remercier profondément aussi les membres du jury pour avoir accepté d'évaluer notre travail.

Table des figures

Liste des tableaux

Chapitre 1

Contexte général

1.1 Introduction générale

Les émotions sont décrites comme des sentiments intenses dirigés à quelque chose ou quelqu'un en réponse à des événements internes ou externes ayant une signification particulière pour l'individu. Aujourd'hui, l'internet est devenu un moyen clé par lequel les gens expriment leurs émotions, leurs sentiments et leurs opinions. Chaque événement, actualité ou activité dans le monde, est partagé, discuté, publié et commenté sur les réseaux sociaux par des millions de personnes. La capture de ces émotions en texte peut être une source des informations précieuses, qui peuvent être utilisées pour étudier comment les gens réagissent à différentes situations et événements.

Les analystes commerciaux peuvent utiliser ces informations pour suivre les sentiments et les opinions des gens sur leurs produits. Les chefs d'entreprise peuvent analyser la vision globale des personnes réponse à leurs actions ou événements et s'adapter avec cette vision. Ainsi, les analystes de la santé peuvent étudier les sautes d'humeur des individus ou masses à différents moments de la journée ou en réponse à certains événements. Ces informations peuvent également être utilisé pour formuler l'état mental d'un individu, étudiant son activité sur une période de temps, et éventuellement détecter les risques de dépression.

Le contexte de notre projet est de concevoir et réaliser une application d'analyse des sentiments des tweets en temps réel.

Ce rapport présente l'ensemble d'étapes suivies pour développer la solution. Il se divise en 5 chapitres organisés comme suit :

- Le premier chapitre "Contexte général" discute l'état de l'art et présente le contexte général du projet.
- Le deuxième chapitre "Étude théorique et choix techniques" contient une étude des solutions techniques existantes et discute les différents choix techniques éxplorés dans notre projet.
- Le troisième chapitre "Analyse des besoins et spécifications" détermine les différents besoins de notre application et traite son aspect conceptuel.
- Le quatrième chapitre "Réalisation" illustre les différentes étapes de la réalisation de l'application.
- En conclusions, nous discutons les difficultés rencontrées lors de la réalisation de notre solution et les différents perspectives d'améliorations possibles.

1.2 Les solutions éxistantes :

Pour mieux expliquer le contexte général de notre projet, on commence par présenter quelques outils éxistantes au marché :

- **Enginuity** : C'est une application Web qui fonctionne différemment de la plupart des outils d'analyse de sentiment libre disponibles. Au lieu d'interroger directement les tweets liés à un certain mot clé, Enginuity permet de rechercher des actualités récentes sur le mot clé. Ensuite l'outil interroge Twitter pour calculer le nombre de fois que l'histoire a été partagée. Il analyse

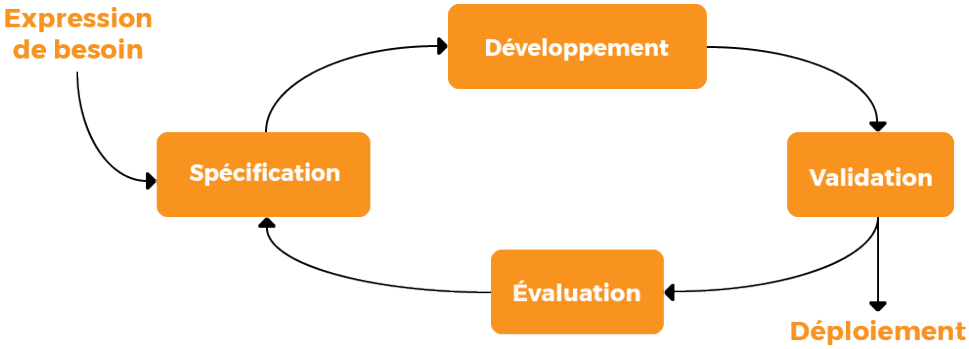

Fig. 1.1 – **Différentes étapes du cycle itératif**

également si le sentiment de ces partages est positif ou négatif et donne une note de sentiment globale.

- **Steamcrab** : C'est aussi une application Web qui se concentre sur les recherches de mots clés et analyse les tweets selon une échelle bipolaire (positive et négative), comme on peut visualiser ces résultats sur un diagramme de dispersion (scatter plot) ou un diagramme circulaire (pie chart).
- **MeaningCloud** : est une API pour l'analyse de texte, y compris l'analyse des sentiments. L'un des avantages de MeaningCloud est que l'API prend en charge un certain nombre d'opérations d'analyse de texte en plus de la classification des sentiments (l'extraction de sujet, la classification grammaticale ou lexicale de texte...).
- **NCSU Tweet Sentiment Visualization App** (Application Web) : l'un des outils gratuits les plus robustes et hautement fonctionnels pour l'analyse des sentiments sur Twitter. Cette application est facile à utiliser : on doit juste entrer un mot clé et elle extrait automatiquement les tweets récents en nous offrant plusieures options à utiliser (la possibilité de séléctionner des tweets individuels sur le "scatter plot" pour les identifier dans le spectre émotionnel). Ainsi, cette application regroupe automatiquement les tweets en sujets connexes en utilisant des algorithmes d'apprentissage automatique, elle combine alors 'analyse des sentiments avec le regroupement des sujets pour nous aider à comprendre ce que les gens pensent des sujets particuliers.

1.3 Présentation générale du projet :

Notre projet consiste à concevoir et développer une application desktop qui nous permet de rechercher les tweets contenants un mot clé ou une phrase donnés en temps réel, obtenir des statistiques générales sur ces tweets et tracer les changements de leurs sentiments, sauvegarder les graphes tracées ainsi que les tweets et ses analyses sur le disque dur.

1.4 Organisation du projet :

Pour la réalisation de notre projet, on a choisi d'adapter le processus itératif comme méthodologie du travail grâce à la souplesse de cette méthode.

1.4.1 Processus itératif :

Ce processus se décompose en 6 étapes (Figure 1.1).

Expression du besoin :

La première réunion avec notre encadrant M. Mustapha AMEUR nous a permis de définir les différents besoins :

- Un outil d'analyse des sentiments à partir des tweets où l'utilisateur peut voir un score de positivité et de négativité de chaque tweet, en traçant graphiquement les variations selon le temps.

Ça nous a permis aussi de définir les besoins non fonctionnels concernant notre système :

- L'ergonomie : l'application offre une interface conviviale et facile à utiliser.

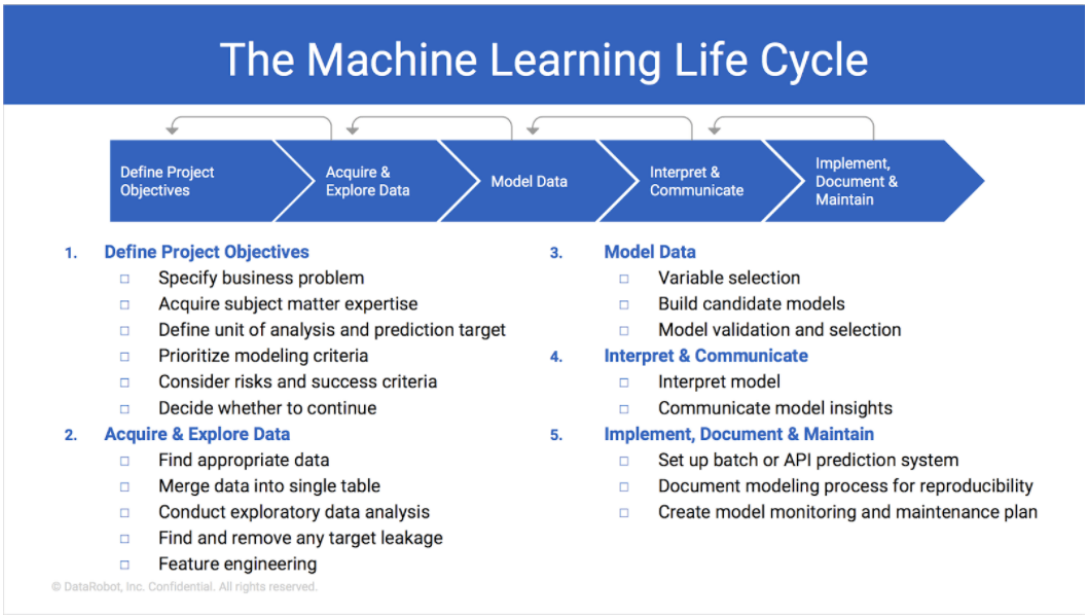

Fig. 1.2 – **Exemple d'étapes d'un projet en machine learning**

- Le code doit être clair et modulaire pour permettre des futures évolutions ou améliorations. Et d'être paramétrable le plus possible afin d'assurer une simplicité lors changements sans être obliger de modifier le code source.

Spécification :

Dans cette étape, nous avons choisi les différents outils et technologies qui vont nous aider à satisfaire les besoins définis dans la première étape :

- La conception et le développement d'une application desktop d'analyse des sentiments des tweets en temps réel en Python à l'aide de : Python-twitter, Vader, nltk, TextBlob, PyQt5. Comme on a décidé de former un modèle Naive Bayes basé sur notre propre "dataset".

Développement :

Dans cette étape, nous avons réalisé en concrèt ce qui a été définit en essayant de respecter un exemple d'un ensemble d'étapes de réalisation d'un projet en machine learning (Figure 1.2) :

Validation :

Ensemble de test permettant d'assurer que le besoin initialement formulé est comblé.

Évaluation :

Cette étape nous a permis de faire le point sur les problèmes rencontrés lors des trois précédentes étapes, chose qui nous a aidé à trouver une solution pour accélerer le modèle entrainé (Naive Bayes) après quelques évaluations.

Déploiement :

les livrables validés sont déployés et prêts à être utilisé.

1.5 Conclusion :

Dans ce chapitre, nous avons présenter l'idée générale et le contexte de notre projet, comme on a détaillé la méthodologie du travail adaptée lors de la réalisation du projet.

Chapitre 2

Étude théorique et conception

2.1 Étude préliminaire :

Les méthodes existantes pour l'analyse des sentiments peuvent être regroupées en deux catégories principales : méthodes basées sur la connaissance (knowledge-based), ou sur l'apprentissage automatique (machine learning).

2.1.1 Méthodes basées sur la connaissance :

Dans les méthodes basées sur la connaissance, également appelé classification des sentiments basée sur le lexique, l'objectif est de construire ou utiliser des lexiques de mots de sentiments existants avec des étiquettes de sentiments indiquées pour les mots ou les phrases dans le texte. La classification du texte est définie par des règles ; une fonction sur les mots, comme la somme des polarités des mots [Taboada et al., 2011]. Cette approche ne nécessite pas toute formation (autre que la formation d'un lexique, si nécessaire).

Hu, Minqing et Bing Liu [Hu and Liu, 2004] ont construit un lexique, utilisant uniquement WordNet (base de données lexicale) et une liste d'adjectifs de semences étiquetés. Cette liste contient uniquement des adjectifs positifs (par exemple, grand, étonnant, agréable, cool) et des adjectifs négatifs (par exemple, mauvais, ennuyeux). Leur méthode récupère et étiquette automatiquement les synonymes (même polarité) et antonymes (polarité opposée). Ce processus permet à la liste de devenir un lexique. Un inconvénient de cette approche est qu'elle n'est applicable que dans les langues où WordNet est disponible. Dans tous les cas, la méthode basée sur les connaissances peut être difficile en raison du bruit dans les données textuelles, tandis que la création manuelle de règles pour combiner des informations sur les mots obtenus à partir des lexiques de sentiments prend du temps et des efforts.

2.1.2 Méthodes basées sur l'apprentissage automatique :

D'autre part, l'apprentissage automatique nécessite de former un modèle pour prédire la polarité du texte. Le modèle est formé avec des messages texte, étiquetés pour leur sentiment et représentés comme des vecteurs de fonctionnalités. Ce dernier nécessite classiquement un prétraitement de texte à l'aide d'outils de traitement de langage comme NLTK6. Le prétraitement du texte implique principalement la tokenisation, le stemming le balisage et éventuellement l'analyse du texte. La sélection des caractéristiques appropriées à partir des données est cruciale et s'est révélée être un problème majeur et est toujours un objectif clé pour les chercheurs.

Des travaux antérieurs sur l'analyse des sentiments ont exploité des méthodes bien connues d'apprentissage automatique (machine learning) supervisé, telles que Naive Bayes [Martinez-Arroyo and Sucar, 2006], SVMs [Vinet and Zhedanov, 2011, Ho, 1995, Wahid et al., 2017]. Des travaux plus récents utilisent des modèles d'apprentissage profond (deep learning) [Goldberg, 2017], en particulier les réseaux neuronaux récurrents (RNN) et le réseau neuronal convolutionnel (CNN). Cette thèse porte également sur les modèles d'apprentissage profond (Figure 2.1).

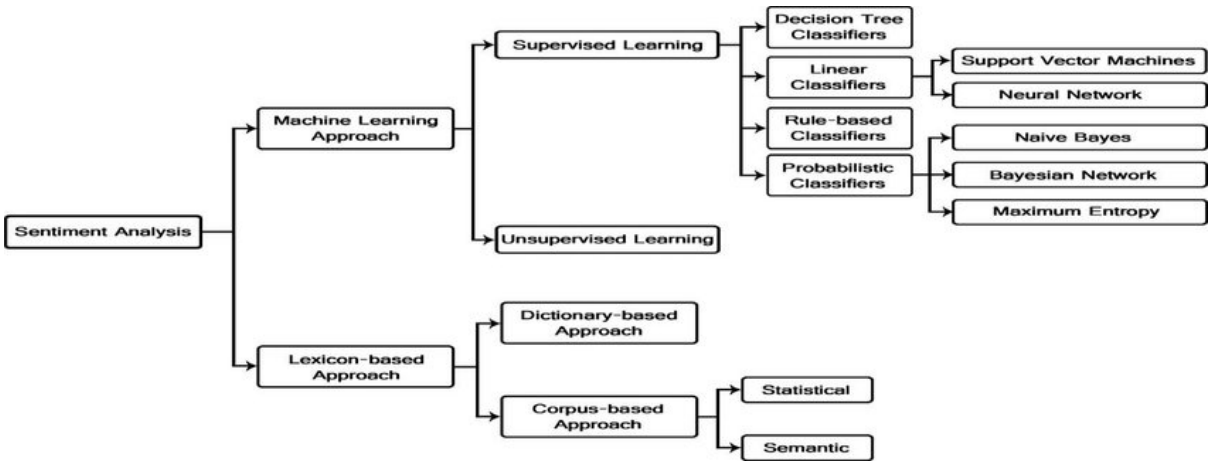

Fig. 2.1 – **Description des méthodes d'analyse des sentiments**

2.1.3 Limites des solutions existantes :

Certaines limitations des solutions existantes de l'analyse des sentiments :

- **Pour les méthodes basées sur la connaissance :**
  - Elles peuvent être difficile en raison du bruit dans les données textuelles, tandis que la création manuelle de règles pour combiner des informations sur les mots obtenus à partir des lexiques de sentiments prend du temps et des efforts.
  - L'approche est qu'elle n'est applicable que dans les langues où le dictionnaire (La base de données lexicale) est disponible.
  - Elles nécessitent de puissantes ressources linguistiques pour extraire des informations à partir des mots, chose qui n'existe pas toujours.
- **Pour les méthodes basées sur l'apprentissage automatique :**
  - La sélection des caractéristiques appropriées à partir des données est cruciale et s'est révélée être un problème majeur et est toujours un objectif clé pour les chercheurs.

2.2 NLP (Natural Language Processing) :

Le traitement du langage naturel est un sous-domaine de la linguistique, l'informatique, l'ingénierie de l'information et de l'intelligence artificielle (Figure 2.2) concerné par les interactions entre les ordinateurs et les langues (naturelles) humaines, en particulier comment programmer des ordinateurs pour traiter et analyser de grandes quantités de données en langage naturel.

Les défis du traitement du langage naturel impliquent fréquemment la reconnaissance vocale, la compréhension du langage naturel et la génération du langage naturel (Figure 2.3).

Ce qui suit est une liste des tâches les plus recherchées dans le traitement du langage naturel. Certaines de ces tâches ont des applications directes dans le monde réel, tandis que d'autres servent généralement de sous-tâches qui sont utilisées pour aider à résoudre des tâches plus importantes.

Bien que les tâches de traitement du langage naturel soient étroitement liées, elles sont fréquemment subdivisées en catégories pour plus de commodité :

2.2.1 Syntaxe :

- **Induction grammaticale :** La Génération d'une grammaire formelle qui décrit la syntaxe d'un langage [Klein and Manning, 2002].
- **Lemmatisation :** La tâche de supprimer uniquement les terminaisons flexionnelles et de renvoyer la forme de dictionnaire de base d'un mot qui est également connu sous le nom de lemme.
- **Segmentation morphologique :** La séparation des mots en morphèmes individuels et identifiez la classe des morphèmes. La difficulté de cette tâche dépend fortement de la complexité de la morphologie (c'est-à-dire la structure des mots) de la langue considérée.

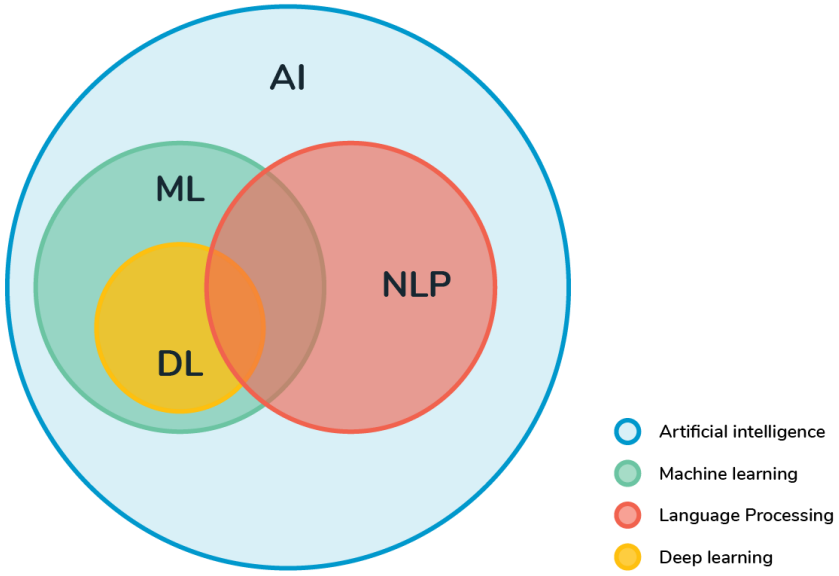

Fig. 2.2 – **NLP est une branche de l'intelligence artificielle**

- **Segmentation des mots :** La séparation d'un morceau de texte continu en mots séparés. Pour une langue comme l'anglais, c'est assez trivial, car les mots sont généralement séparés par des espaces. Cependant, certaines langues écrites ne marquent pas les limites des mots de cette manière, et dans ces langues, la segmentation du texte est une tâche difficile qui nécessite la connaissance du vocabulaire et de la morphologie des mots dans la langue.
- **Catégorie grammaticale (Part-of-speech tagging) :** Cela implique d'identifier la catégorie grammaticale pour chaque mot.
- **Analyse (Parsing) :** La détermination d'une arbre d'analyse (analyse grammaticale) d'une phrase donnée. La grammaire des langues naturelles est ambiguë et les phrases typiques ont de multiples analyses possibles. Peut-être de façon surprenante, pour une phrase typique, il peut y avoir des milliers d'analyses potentielles (dont la plupart sembleront complètement absurdes pour un humain).
- **Rupture de phrase :** Étant donné un morceau de texte, on trouve les limites de la phrase. Ces limites sont souvent marquées par des points ou d'autres signes de ponctuation, mais ces mêmes caractères peuvent servir à d'autres fins (par exemple, marquer des abréviations).
- **Stemming :** Cela implique de couper les mots à leur forme racine.
- **Extraction terminologique :** Le but de l'extraction terminologique est d'extraire automatiquement les termes pertinents d'un corpus donné.

2.2.2 Sémantique :

- **Sémantique lexicale :** La signification informatique des mots individuels dans leur contexte.
- **Sémantique distributionnelle :** L'apprentissage des représentations sémantiques à partir des données.
- **Reconnaissance des entités nommées :** Il s'agit de déterminer les parties d'un texte qui peuvent être identifiées et classées en groupes prédéfinis. Par exemple les noms de personnes et les noms de lieux.
- **Désambiguïsation du sens des mots :** Beaucoup de mots ont plus d'un sens ; nous devons sélectionner le sens qui est le plus convenable au contexte. Pour ce problème, on reçoit généralement une liste de mots et de sens de mots associés, par exemple à partir d'un dictionnaire ou d'une ressource en ligne telle que WordNet.
- **Extraction de relations :** Étant donné un morceau de texte, on identifie les relations entre les entités nommées.
- **Génération du langage naturel :** Il s'agit d'utiliser des bases de données pour dériver des

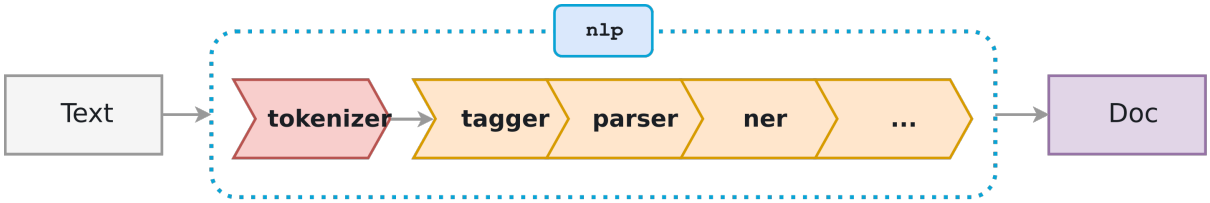

Fig. 2.3 – Processus NLP

intentions sémantiques et les convertir en langage humain.

2.2.3 Discours :

- **Synthèse automatique :** La production d'un résumé lisible d'un morceau de texte.
- **Résolution de coréférence :** Étant donné une phrase ou un morceau de texte, on détermine quels mots (mentions) font référence aux mêmes objets (entités).

2.2.4 Parole :

- **Reconnaissance de la parole :** À partir d'un extrait sonore d'une ou de plusieurs personnes parlant, on détermine la représentation textuelle du discours.
- **Segmentation de la parole :** Étant donné un extrait sonore d'une ou de plusieurs personnes parlant, on le sépare en mots. Une sous-tâche de reconnaissance vocale et généralement regroupée avec elle.
- **Synthèse de la parole à partir du texte (Text-to-speech)** Étant donné un texte, on transforme ses unités et on produit une représentation parlée. La synthèse vocale peut être utilisée pour aider les gens avec une basse vision [Yi and Tian, 2011].

2.3 Algorithme Naive Bayes :

L'algorithme Naive Bayes est un algorithme de classification probabiliste simple qui s'appuie sur le théorème de Bayes. Naive Bayes a des hypothèses d'indépendance fortes (naïves) entre les caractéristiques. En termes simples, un classificateur Naive Bayes suppose que la présence d'une caractéristique particulière dans une classe n'est pas liée à la présence de toute autre caractéristique. Par exemple, un ballon peut être considéré comme un ballon de football s'il est dur, rond et d'environ sept pouces de diamètre. Même si ces caractéristiques dépendent les unes des autres ou de l'existence des autres caractéristiques, Bayes naïf pense que toutes ces propriétés contribuent indépendamment à la probabilité que ce ballon soit un ballon de football, c'est pourquoi il est connu comme naïf.

Les modèles Naive Bayes sont faciles à construire. Ils sont également très utiles pour de très grands ensembles de données. Bien que les modèles Bayes naïfs soient simples, ils sont connus pour surpasser même les modèles de classification les plus sophistiqués. Parce qu'ils nécessitent également un temps de formation relativement court, ils constituent une bonne alternative pour une utilisation dans les problèmes de classification.

Résuméement, Bayes naïf est un modèle de probabilité conditionnel : étant donné une instance de problème à classer, représentée par un vecteur $x = x_1, ..., x_n$ représentant n fonctionnalités (variables indépendantes), il assigne à cette instance des probabilités $p(C_k|x_1, ..., x_n)$ pour chaque K résultats ou classes possibles ou classes C_k [Murty and Devi, 2011].

Le problème avec la formulation ci-dessus est que si le nombre d'entités n est grand ou si une entité peut prendre un grand nombre de valeurs, alors baser un tel modèle sur des tables de probabilités est irréalisable. Nous reformulons donc le modèle pour le rendre plus maniable. En utilisant le théorème

de Bayes, la probabilité conditionnelle peut être décomposée comme :

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)} \tag{2.1}$$

En pratique, il n'y a d'intérêt que pour le numérateur de cette fraction, car le dénominateur ne dépend pas de C et les valeurs des traits x_i sont données, de sorte que le dénominateur est effectivement constant. Le numérateur est équivalent au modèle de probabilité conjoint $p(C_k, x_1, ..., x_n)$ qui peut être réécrite comme suit, en utilisant la règle de chaîne de la définition de la probabilité conditionnelle :

$$\begin{aligned} p\left(C_k, x_1, \ldots, x_n\right) &= p\left(x_1, \ldots, x_n, C_k\right) \\ &= p\left(x_1 | x_2, \ldots, x_n, C_k\right) p\left(x_2, \ldots, x_n, C_k\right) \\ &= p\left(x_1 | x_2, \ldots, x_n, C_k\right) p\left(x_2 | x_3, \ldots, x_n, C_k\right) p\left(x_3, \ldots, x_n, C_k\right) \\ &= \cdots \\ &= p\left(x_1 | x_2, \ldots, x_n, C_k\right) p\left(x_2 | x_3, \ldots, x_n, C_k\right) \ldots p\left(x_{n-1} | x_n, C_k\right) p\left(x_n | C_k\right) p\left(C_k\right) \end{aligned} \tag{2.2}$$

Maintenant, les hypothèses d'indépendance conditionnelle "naïve" entrent en jeu : supposons que toutes les fonctionnalités de x sont mutuellement indépendantes, conditionnelles à la catégorie C_k, dans cette hypothèse, $p\left(x_i | x_{i+1}, \ldots, x_n, C_k\right) = p\left(x_i | C_k\right)$

Ainsi, le modèle commun peut être exprimé comme :

$$\begin{aligned} p\left(C_k | x_1, \ldots, x_n\right) &\propto p\left(C_k, x_1, \ldots, x_n\right) \\ &= p\left(C_k\right) p\left(x_1 | C_k\right) p\left(x_2 | C_k\right) p\left(x_3 | C_k\right) \cdots \\ &= p\left(C_k\right) \prod_{i=1}^{n} p\left(x_i | C_k\right) \end{aligned} \tag{2.3}$$

où $\propto$ dénote la proportionnalité.

Cela signifie que sous les hypothèses d'indépendance ci-dessus, la distribution conditionnelle sur la variable de classe C est :

$$p\left(C_k | x_1, \ldots, x_n\right) = \frac{1}{Z} p\left(C_k\right) \prod_{i=1}^{n} p\left(x_i | C_k\right)$$

où $Z = p(\mathbf{x}) = \sum_k p\left(C_k\right) p\left(\mathbf{x} | C_k\right)$ $x_1, \ldots, x_n$ c'est-à-dire une constante si les valeurs des variables d'entité sont connues.

Construire un classificateur à partir du modèle de probabilité : Jusqu'à présent, la discussion a dérivé le modèle d'entité indépendant, c'est-à-dire le modèle de probabilité naïf de Bayes. Le classificateur naïf de Bayes combine ce modèle avec une règle de décision. Une règle courante consiste à choisir l'hypothèse la plus probable ; il s'agit de la règle de décision maximale a posteriori ou MAP(maximum a posteriori). Le classificateur Bayes, est la fonction qui attribue une étiquette de classe $\hat{y} = C_k$ pour certains k comme suit :

$$\hat{y} = \underset{k \in \{1, \ldots, K\}}{\operatorname{argmax}} p\left(C_k\right) \prod_{i=1}^{n} p\left(x_i | C_k\right)$$

2.4 Vader :

Vader (Valence Aware Dictionary and sentiment Reasoner) est un lexique et un outil d'analyse des sentiments basé sur des règles, qui est spécifiquement adapté aux sentiments exprimés dans les médias sociaux. Vader utilise une combinaison d'un lexique de sentiments qui est une liste de caractéristiques lexicales généralement étiquetées selon leur orientation sémantique comme positives ou négatives.

Vader s'est avéré très efficace dans le traitement de textes de médias sociaux, de critiques de films et de produits. Cela est dû au fait que Vader ne parle pas seulement du score de positivité et de

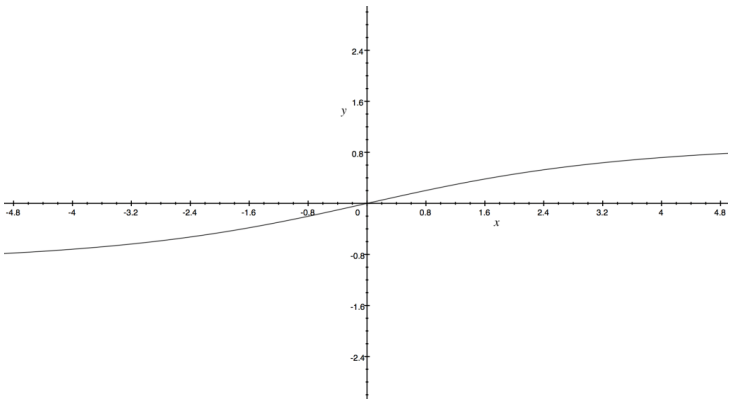

Fig. 2.4 – **Normalisation de Vader**

négativité, mais nous indique également à quel point un sentiment est positif ou négatif (compound) [Hutto and Gilbert, 2014].

2.4.1 Quantifier l'émotion d'un mot :

L'analyse des sentiments Vader s'appuie principalement sur un dictionnaire qui mappe les caractéristiques lexicales aux intensités émotionnelles appelées scores des sentiments. Le score de sentiment d'un texte peut être obtenu en résumant l'intensité de chaque mot dans le texte. Et par caractéristiques lexicales, on veut dire tout ce que nous utilisons pour la communication textuelle (mots, "emojis" au cas de l'analyse des tweets par exemple, acronymes...)

L'intensité d'émotion ou le score de sentiment est mesuré sur une échelle de -4 à +4, où -4 est le plus négatif et +4 est le plus positif. Le point médian 0 représente un sentiment neutre.

Vader construit son dictionnaire en utilisant les évaluateurs humains d'Amazon Mechanical Turk, et pour contrer la subjectivité des évaluateurs (notions différentes de ce qui est positif ou négatif), les créateurs de Vader ont utilisé non seulement un, mais un certain nombre d'évaluateurs humains et ont fait la moyenne de leurs notes pour chaque mot. Cela repose sur le concept de la sagesse de la foule (the wisdom of the crowd).

2.4.2 Quantifier l'émotion d'une phrase :

L'analyse de sentiment Vader renvoie un score de sentiment compris entre -1 et 1, du plus négatif au plus positif en sommant les scores de sentiment de chaque mot de la phrase existant dans le dictionnaire Vader avant de normaliser cette somme pour le mapper à une valeur comprise entre -1 et 1. La normalisation utilisée par [Hutto and Gilbert, 2014] :

$$\frac{x}{\sqrt{x^2+\alpha}}$$

x est la somme des scores de sentiment des mots constitutifs de la phrase et α est un paramètre de normalisation que nous fixons à 15. La normalisation est représentée graphiquement dans la figue **??**.

Nous voyons ici que lorsque x grandit, il devient de plus en plus proche de -1 ou 1. Pour un effet similaire, s'il y a beaucoup de mots dans le document auquel vous appliquez l'analyse des sentiments Vader, on obtient un score proche de -1 ou 1. Ainsi,l'analyse des sentiments Vader fonctionne mieux sur les documents courts, comme les tweets et les phrases, pas sur les gros documents.

2.5 Conception du système :

Dans cette étape, nous abordons la partie conception du projet, dans laquelle, nous détaillons les différents éléments de conception, en commençant d'abord par spécifier les besoins fonctionnels de notre système, établir une architecture générale puis une modélisation statique du système à travers un diagramme de classes.

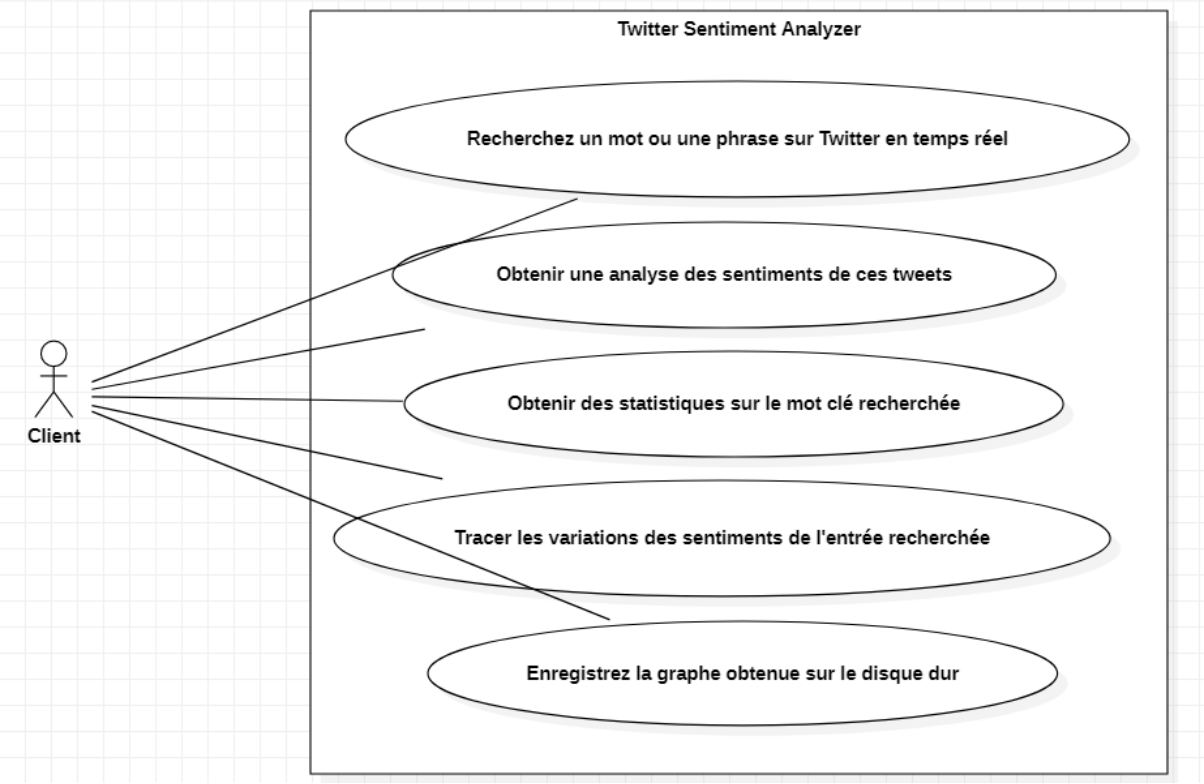

Fig. 2.5 – **Diagramme de cas d'utilisation**

2.5.1 Spécification des besoins fonctionnels :

La spécification de ces besoins va nous permettre d'avoir une meilleure approche des utilisateurs, des fonctionnalités et de la relation entre les deux. Elle sera alors sous forme d'un diagramme de cas d'utilisation où on va procéder par l'identification des acteurs agissant sur notre système et les différentes cas d'utilisation (Figure 2.5).

2.5.2 Architecture générale de la solution :

Pour réaliser notre système, on a choisi à suivre une architecture adapté à la nature de notre projet que nous représentons dans ce diagramme (Figure 2.6).

Comme on remarque dans le diagramme ci-dessus, on peut diviser le fonctionnement de notre application en cinque grandes parties :

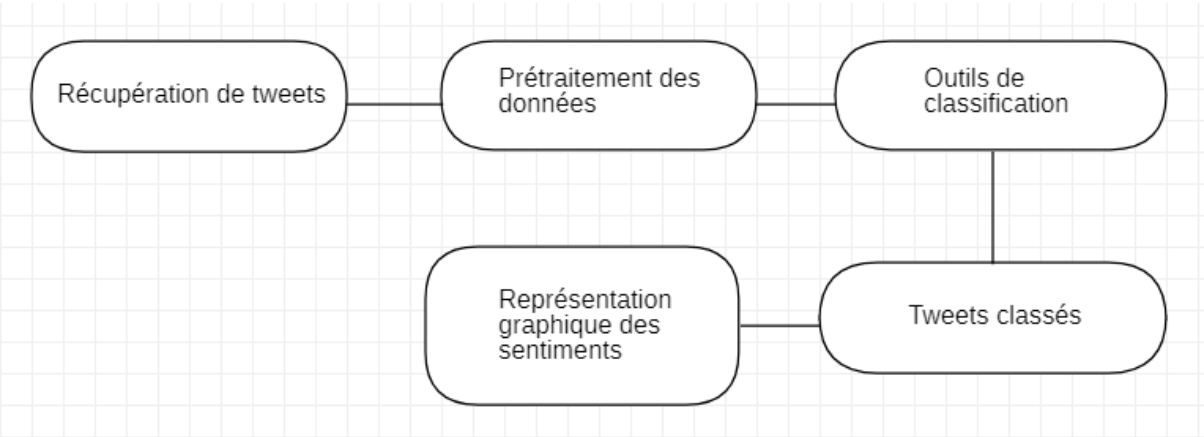

Fig. 2.6 – **Architecture du système**

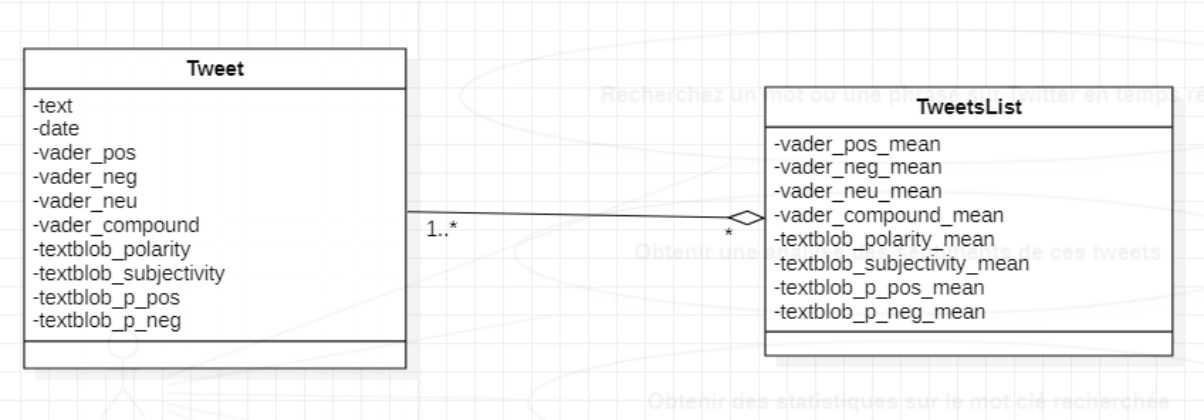

Fig. 2.7 – **Diagramme de classes**

- **Récupération de tweets :** La recherche et l'extraction des tweets contenant le mot clé entré par l'utilisateurs.
- **Prétraitement des données :** Cette partie consiste à appliquer les différentes tâches et techniques de "NLP" pour traiter les tweets récupérés.
- **Outils de classification :** Définir les méthodes suivies pour l'analyse de sentiments et la classification des tweets.
- **Tweets classés :** Classer ces tweets par des labels (positif ou négatif par exemple) et donner une évaluation générale sur l'ensemble de tweets récupérés.
- **Représentation graphique des sentiments :** Représentation graphique des sentiments de tweets selon les méthodes d'analyse choisie.

2.5.3 Diagramme de classes :

Le diagramme de classes est une modélisation statique du système en termes de classes et de relations entre ces classes. Son intérêt réside dans la modélisation des entités du système d'information.

Vu la nature de notre projet, nous avons décidé de créer deux entités illustrées dans le diagramme dans la figure 2.7.

- **Tweet :** On utilise cette classe pour stocker les différentes informations liées à chaque tweet récupérés.
- **TweetsList :** On utilise cette classe pour stocker les informations liées à l'ensemble de tweets récupérés.

2.6 Conclusion :

Dans ce chapitre, nous avons effectué une étude théorique où on a exploré les différents concepts utilisé pour la réalisation du projet, comme on a abordé le côté conceptuel du projet.

Chapitre 3

Les choix techniques

3.1 Solution détaillée :

Notre solution consiste à concevoir et développer une application d'analyse des sentiments des tweets en temps réel en utilisant les outils et les librairies suivantes :

- Chercher les tweets contenant le mot clé entré, en utilisant **python-twitter**, la librairie qui fournit une interface Python pure pour utiliser l'API Twitter.
- Analyser ces tweets à travers des models fournis par **TextBlob**, **Vader**, **nltk**, et par un modèle Naive Bayes formé par nous-même, pour obtenir des statistiques sur le mot clé entré.
- Tracer les changements des sentiments de ces tweets.
- Donner la possibilité de sauvegarder les graphes tracées, ainsi que les tweets et ses analyses, sur le disque dur sous plusieurs formats (PNG, JPEG...).

L'interaction avec l'utilisateur se fait via une interface graphique réalisée à l'aide de **PyQt5** (l'ensemble de liaisons Python pour Qt), où s'affichent les tweets, les différentes statistiques ainsi que les graphes tracées.

3.2 Environnement logiciel :

Nous avons intérêt à mettre en place une architecture rigoureuse, de manière à garantir la maintenabilité, l'évolutivité et l'exploitabilité de notre application.

3.2.1 Python :

La nature de notre projet (Analyse de sentiments) nous a poussé à choisir Python pour le développement de notre application surtout qu'il est le langage le plus populaire pour les projets d'intelligence artificielle, "machine learning", et le "deep learning" : [Luashchuk, 2019]

Python nous offre alors :

Un grand écosystème de bibliothèques :

Un grand choix de bibliothèques est l'une des principales raisons pour lesquelles Python est le langage de programmation le plus populaire utilisé pour l'intelligence artificielle et le "machine learning", surtout que ce dernier nécessite un traitement continu des données, et les bibliothèques Python nous permettent d'accéder, de gérer et de transformer les données.

On mentionne comme exemple : Scikit-learn, Pandas, TensorFlow, Matplotlib, NLTK, etc.

Flexibilité :

Il offre l'option pour choisir entre utiliser POO ou la méthode fonctionnelle, et il n'est pas non plus nécessaire de recompiler le code source, les développeurs peuvent implémenter toutes les modifications

et voir rapidement les résultats, comme ils peuvent combiner Python et d'autres langages pour atteindre leurs objectifs.

Lisibilité :

Python est très facile à lire pour que chaque développeur Python puisse comprendre le code de ses pairs et le modifier, le copier ou le partager. Il n'y a pas de confusion, d'erreurs ou de paradigmes conflictuels, ce qui conduit à un échange plus efficace d'algorithmes, d'idées et d'outils.

3.2.2 Jupyter Notebook :

Le "notebook" étend l'approche basée sur la console à l'informatique interactive dans une nouvelle direction qualitative, en fournissant une application Web adaptée à la capture de l'ensemble du processus de calcul : développement, documentation et exécution de code, ainsi que communication des résultats. Le bloc-notes Jupyter combine deux composants :

- **Une application Web :** Un outil basé sur un navigateur pour la création interactive de documents qui combinent du texte explicatif, des mathématiques, des calculs et leur output media.
- **Documents "notebook" :** Une représentation de tout le contenu visible dans l'application Web, y compris les entrées et sorties des calculs, le texte explicatif, les mathématiques, les images et les représentations media des objets.

3.2.3 Google Colab :

Google a fourni un service cloud gratuit basé sur les "Jupyter Notebooks" qui prend en charge le GPU gratuit. Google Colab nous permet de développer des applications de "deep learning" à l'aide de bibliothèques populaires telles que PyTorch, TensorFlow, Keras et OpenCV.

On peut créer des "notebooks" dans Colab, les télécharger, les stocker, les partager, monter Google Drive et utiliser tout ce qu'on y a stocké, importer la plupart de nos répertoires préférés, télécharger des "notebooks" directement depuis GitHub, téléchargez des fichiers Kaggle, et plein d'autre fonctionnalités [Bonner, 2019].

3.3 Conclusion :

Dans ce chpitre, nous avons présenté notre solution envisagée ainsi que l'environnement logiciel dans laquelle nous avons réalisé notre projet.

Chapitre 4

Réalisation

4.1 Introduction

Pour faciliter le travail, nous avons utilisé Pyqt5 pour l'implémentation d'une GUI, des modèles pré-formés pour nous aider à obtenir des résultats rapides, et nous avons choisi de former nous-mêmes un modèle "Naive Bayes" sur un ensemble de tweets de 2009 (Annexe A). L'ensemble de données est composé de plus de 1,6 millions de tweets.

4.2 Notre implementation de Naive Bayes :

Pour former notre modèle Bayes naïf, nous importons les données brutes de notre base de données CSV comme dans le tableau 4.1.

Ces tweets bruts doivent être pré-traités dans d'autres pour que les Bayes naïfs les utilisent comme un ensemble d'entraînement. Nous ouvrons les tweets chez les pandas pour faciliter la manipulation. Nous choisissons uniquement ce qui nous intéresse dans les colonnes : le texte et la partition.

L'ensemble de données est ordonné en deux catégories, positives et négatives. Nous avons donc dû mélanger l'ensemble de données pour créer une distribution aléatoire des entrées et obtenir un ensemble équilibré pour la formation et les tests.

Le prétraitement des tweets devait se faire en plusieurs étapes :

- Convertir tous les mots en minuscules pour assurer l'homogénéité des tweets et du dictionnaire ;
- Supprimer les URL et les remplacer par le mot clé "URL" pour préserver l'intégrité des tweets ;
- Supprimer les marques et les remplacer par le mot-clé "AT_USER" ;
- Retirez le signe du hashtag ;
- Tokenize les tweets pour supprimer les caractères répétés puis les diviser en mots ;
- Tout d'abord, nous avons dû supprimer les mots inutiles basés sur la bibliothèque "stopwords" de NLTK ;

Après le prétraitement de tous les tweets, nous avons divisé l'ensemble de données en un ensemble d'apprentissage de 95% de tous les tweets et de 5% pour le test.

score	texte
0	@switchfoot http ://twitpic.com/2y1zl - Awww, t...
0	is upset that he can't update his Facebook by...
0	@Kenichan I dived many times for the ball. Man...
0	my whole body feels itchy and like its on fire
0	@nationwideclass no, it's not behaving at all....

Table *4.1* – Données brutes pour former notre modèle

Sur la base des tweets traités, nous construisons le dictionnaire de vocabulaire, qui sera utilisé pour faire correspondre les mots avec leur index correspondant, pour permettre au modèle de s'entraîner sur ces index.

La formation sur les serveurs Tensor Processing Unit sur Google Colab se bloquait si nous utilisions plus de 100 000 tweets. Nous avons donc opté pour la formation du modèle sur 5000 tweets de base afin de permettre au serveur de gérer le mot et de ne pas prendre plus de 6 heures.

Nous avons obtenu un résultat respectable pour les paramètres par défaut avec une précision de 84%.

Nous enregistrons les poids et les paramètres du modèle à l'aide de cornichons (pickles) et du vocabulaire dans un fichier CSV utilisé pour la prochaine utilisation.

4.3 Librairies et APIs utilisés :

4.3.1 Python-Twitter :

C'est une bibliothèque qui fournit une interface Python pure pour l'API Twitter. Twitter met en disposition un service qui permet aux gens de se connecter via le Web, la messagerie instantanée et les SMS, Twitter expose alors une API Web et cette bibliothèque a pour but de la rendre encore plus facile à utiliser pour les programmeurs Python [Python-Twitter, 2016].

Twitter API :

À un niveau élevé, les API sont la façon dont les programmes informatiques se «parlent» les uns aux autres pour pouvoir demander et fournir des informations. Pour ce faire, une application logicielle peut appeler ce que l'on appelle un point de terminaison : une adresse qui correspond à un type spécifique d'informations fournis (les points de terminaison (endpoints) sont généralement uniques comme les numéros de téléphone). Twitter permet d'accéder à certaines parties de l'API pour permettre aux utilisateurs de créer un logiciel qui s'intègre à Twitter [TwitterAPI, 2019].

Les données Twitter sont uniques par rapport aux données partagées par la plupart des autres plateformes sociales car elles reflètent les informations que les utilisateurs choisissent de partager publiquement, et cet API nous donne l'accès à ces données. L'API prend également en charge les API qui permettent aux utilisateurs de gérer leurs propres informations Twitter non publiques (par exemple, les messages directs) et de fournir ces informations aux développeurs qu'ils ont autorisés à le faire.

Accéder aux données Twitter : Par défaut, les applications ne peuvent accéder qu'aux informations publiques sur Twitter. Certains points de terminaison (endpoints), tels que ceux chargés d'envoyer ou de recevoir des messages directs, nécessitent des autorisations supplémentaires de la part de l'utilisateur avant de pouvoir accéder à ses informations, et ces autorisations ne sont pas accordées par défaut.

L'API Twitter inclut un large éventail de points de terminaison, qui se répartissent en cinq groupes principaux :

- **Comptes et utilisateurs :** L'API permet aux développeurs de gérer (par programme) le profil et les paramètres d'un compte, de désactiver ou de bloquer les utilisateurs, de gérer les utilisateurs et les abonnés, de demander des informations sur l'activité d'un compte autorisé, et d'autres fonctionnalités.
- **Tweets et réponses :** L'API met les Tweets et les réponses publiques à la disposition des développeurs et permet à ces derniers de publier des Tweets. Les développeurs peuvent accéder aux Tweets en recherchant des mots clés spécifiques ou en demandant un échantillon de Tweets à des comptes spécifiques.
- **Messages directs :** Les points de terminaison de message direct permettent d'accéder aux conversations DM des utilisateurs qui ont explicitement accordé l'autorisation à une application spécifique, ils fournissent un accès limité aux développeurs pour créer des expériences personnalisées sur Twitter. Par exemple pour les comptes qu'elles possèdent ou gèrent, les entreprises peuvent créer ces expériences conversationnelles alimentées par des humains ou des "chatbots" pour communiquer directement avec les clients.
- **Les publicités :** Twitter met à disposition une suite d'API pour permettre aux développeurs d'aider les entreprises à créer et gérer automatiquement des campagnes publicitaires sur Twitter.

Les développeurs peuvent utiliser des Tweets publics pour identifier les sujets et les intérêts, et fournir aux entreprises des outils pour lancer des campagnes publicitaires pour atteindre les divers publics sur Twitter.

- **Outils de l'éditeur et SDK :** L'API fournit aussi des outils aux développeurs et éditeurs de logiciels pour intégrer des boutons de partage et d'autres contenus Twitter sur des pages Web. Ces outils permettent aux développeurs d'intégrer des conversations publiques en direct de Twitter dans leurs plateformes et facilitent le partage d'informations et d'articles de leurs sites avec les utilisateurs.

4.3.2 NLTK :

NLTK (Natural Language Toolkit) est une plate-forme leader pour la construction de programmes Python pour travailler avec des données de langage humain. Il fournit des interfaces faciles à utiliser à plus de 50 corpus et ressources lexicales telles que WordNet, ainsi qu'une suite de bibliothèques de traitement de texte pour la classification, la tokenisation, le stemming, le balisage, l'analyse et le raisonnement sémantique. En autre mot, NLTK aide la machine à analyser, prétraiter et comprendre le texte écrit [Bird et al., 2009].

Tokenization : "Tokenization" est la première étape de l'analyse de texte. Le processus de décomposition d'un paragraphe de texte en morceaux plus petits tels que des mots ou des phrases est appelé "Tokenization". Le jeton (token) est une entité unique qui constitue des blocs de construction pour une phrase ou un paragraphe.

Distribution de fréquence : NLTK nous permet aussi à calculer la fréquence correspondante à chaque mot donné (Figure 4.1).

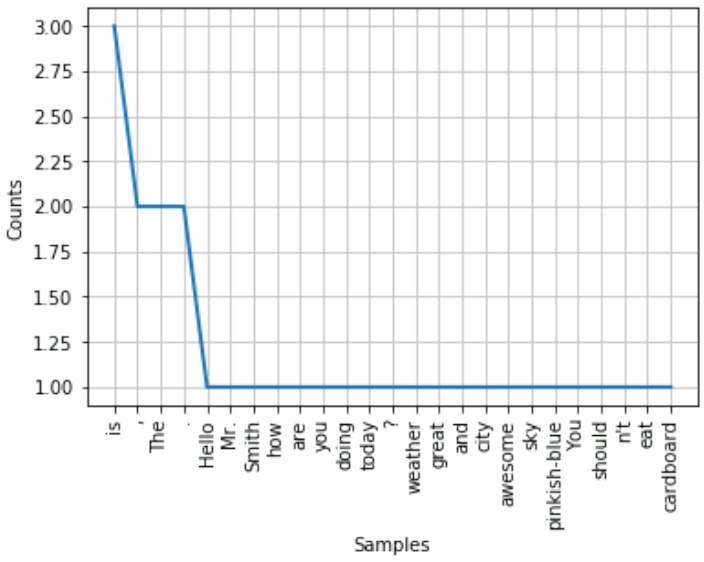

Fig. 4.1 – **Exemple d'une représentation de la distribution de fréquence**

Mots vides (Stopwords) : Pour supprimer les mots vides dans NLTK, on doit créer une liste de mots vides et filtrer la liste de jetons à partir de ces mots.

Normalisation du lexique : La normalisation du lexique considère un autre type de bruit dans le texte. Par exemple, "connexion", "connecté", "connectant" se réduit à un mot commun "se connecter". Il réduit les formes dérivées d'un mot à un mot racine commun.

Stemming : Le "Stemming" est un processus de normalisation linguistique, qui réduit les mots à leur mot racine ou coupe les affixes de dérivation.

Lemmatisation : La lemmatisation réduit les mots à leur mot de base.

Balisage par catégorie grammaticale (POS tagging) : L'identification du groupe grammatical d'un mot donné (Figure 4.2).

1.	CC	Coordinating conjunction	25.	TO	*to*
2.	CD	Cardinal number	26.	UH	Interjection
3.	DT	Determiner	27.	VB	Verb, base form
4.	EX	Existential *there*	28.	VBD	Verb, past tense
5.	FW	Foreign word	29.	VBG	Verb, gerund/present participle
6.	IN	Preposition/subordinating conjunction	30.	VBN	Verb, past participle
7.	JJ	Adjective	31.	VBP	Verb, non-3rd ps. sing. present
8.	JJR	Adjective, comparative	32.	VBZ	Verb, 3rd ps. sing. present
9.	JJS	Adjective, superlative	33.	WDT	*wh*-determiner
10.	LS	List item marker	34.	WP	*wh*-pronoun
11.	MD	Modal	35.	WP$	Possessive *wh*-pronoun
12.	NN	Noun, singular or mass	36.	WRB	*wh*-adverb
13.	NNS	Noun, plural	37.	#	Pound sign
14.	NNP	Proper noun, singular	38.	$	Dollar sign
15.	NNPS	Proper noun, plural	39.	.	Sentence-final punctuation
16.	PDT	Predeterminer	40.	,	Comma
17.	POS	Possessive ending	41.	:	Colon, semi-colon
18.	PRP	Personal pronoun	42.	(	Left bracket character
19.	PP$	Possessive pronoun	43.	)	Right bracket character
20.	RB	Adverb	44.	"	Straight double quote
21.	RBR	Adverb, comparative	45.	'	Left open single quote
22.	RBS	Adverb, superlative	46.	“	Left open double quote
23.	RP	Particle	47.	'	Right close single quote
24.	SYM	Symbol (mathematical or scientific)	48.	”	Right close double quote

Fig. 4.2 – **Différents "POS tags" pour la langue anglaise**

4.3.3 TextBlob :

TextBlob est une bibliothèque Python pour le traitement de données textuelles. Il fournit une API simple pour plonger dans des tâches courantes de traitement du langage naturel (NLP) [Loria, 2018], il nous permet d'analyser les sentiments d'un texte et classer le résultat en deux catégories :

- **Polarité :** est une valeur flottante dans la plage [-1,0 à 1,0] où 0 indique neutre, +1 indique un sentiment très positif et -1 représente un sentiment très négatif.
- **Subjectivité :** est une valeur flottante dans la plage [0,0 à 1,0] où 0 est très objectif et 1 est très subjectif. La phrase subjective exprime certains sentiments personnels, vues, croyances, opinions, allégations, désirs, croyances, soupçons et spéculations alors que les phrases objectives sont factuelles.

TextBlob est en fait construit sur NLTK et Pattern (un autre module de traitement de données textuelles), il fournit une interface facile à utiliser à la bibliothèque NLTK, et nous permet donc d'utiliser une variété de fonctionnalités telles que le balisage par catégorie grammaticale (POS tagging), "Tokenization", "Stemming", Lemmatisation et bien d'autres :

Convertir le texte au singulier et au pluriel : Quand on passe le texte qu'on veut analyser comme paramètre à la classe "TextBlob", TextBlob nous permet de convertir l'objet retourné au singulier ou au pluriel à l'aide de deux méthode : "singularize()" et "pluralize()".

Extraction de phrases nominales : L'extraction de phrases nominales, comme son nom l'indique, fait référence à l'extraction de phrases débutant avec des noms. Pour le faire on doit simplement utiliser les attributs "noun_phrase" sur l'objet TextBlob.

Correction d'orthographe : La correction d'orthographe est l'une des fonctionnalités uniques de TextBlob. Avec la méthode "correct()" de l'objet TextBlob, on peut corriger toutes les fautes d'orthographe dans votre texte.

Traduction de la langue : L'une des capacités les plus puissantes de TextBlob est de traduire d'une langue à une autre. Sur le backend, le traducteur de langue TextBlob utilise l'API Google Translate. Il suffit de passer le texte à l'objet TextBlob puis d'appeler la méthode "translate" sur l'objet. Le code de langue souhaitée comme résultat est transmis en tant que paramètre à la méthode. On peut même détecter la langue d'une phrase en utilisant la méthode "detect_language()".

Classification de texte : TextBlob fournit également des capacités de classification de texte de base. Cependant, TextBlob n'est pas recommandé pour la classification de texte pour les modèles avancés en raison de ses capacités limitées, c'est mieux donc d'utiliser des bibliothèques d'apprentissage automatique telles que Scikit-Learn ou Tensorflow.

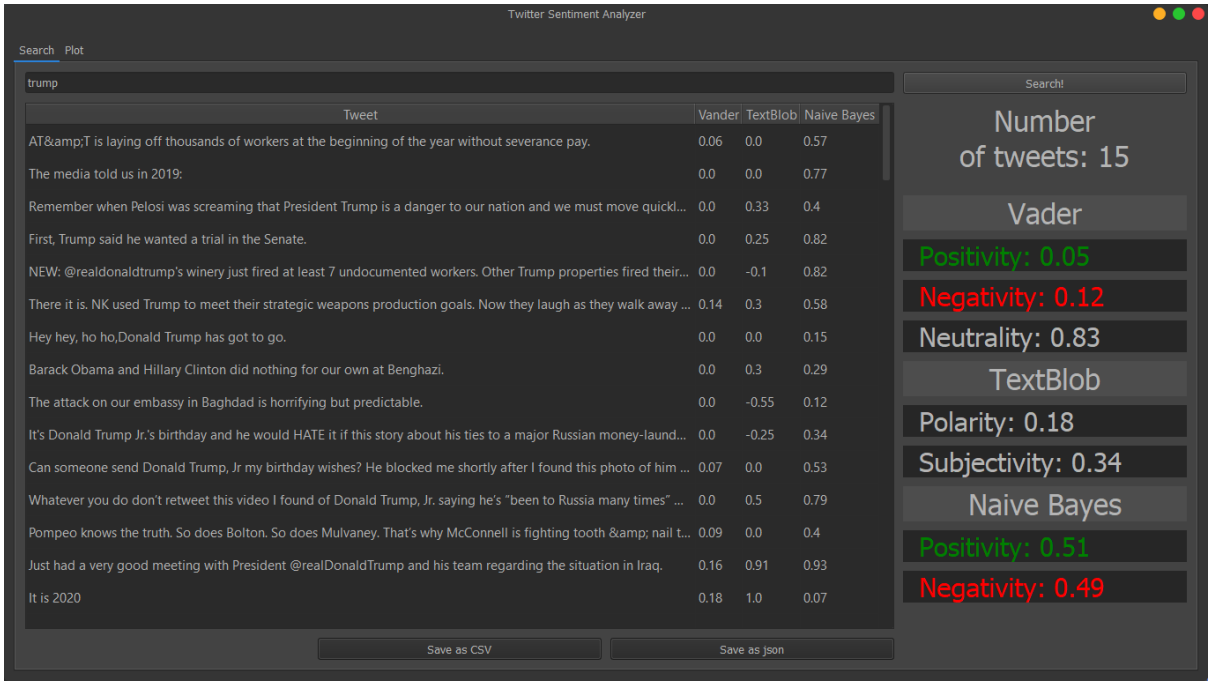

Fig. 4.3 – **fenêtre de recherche**

4.4 Interface graphique

Pour l'interaction du client avec notre système, nous avons choisi d'utiliser Pyqt5 pour implémenter une GUI simple et intuitive composée de deux fenêtres (Figures 4.3 et 4.4). Comme on voit dans la photo ci-dessus, la fenêtre de recherche donne la possibilité au client de :

- Entrer un mot ou une phrase et rechercher (en cliquant sur "search") les tweets qui contiennent le texte entré.
- Visionner le nombre total et les résultats de ces tweets, les scores de chaque tweet, et le score moyen de tous les tweets selon les différents outils d'analyse de sentiments utilisés.
- Enregistrer ses résultats dans le disque dur selon le format choisi (Save as CSV, Save as JSON)

La fenêtre "plot" donne la main au client pour :

- Tracer graphiquement (en cliquant sur "plot") les variations de sentiments selon les moyens d'analyse de sentiments présents.
- Enregistrer ces graphes dans le disque dur (en cliquant sur l'icone d'enregistrement).

4.5 Conclusion

Dans ce chapitre, nous avons détaillé notre implémentation de l'algorithme Bayes Naive, discuté les différentes librairies et APIs choisis pour la réalisation du projet, aussi que les différents écran constituant notre interface graphique.

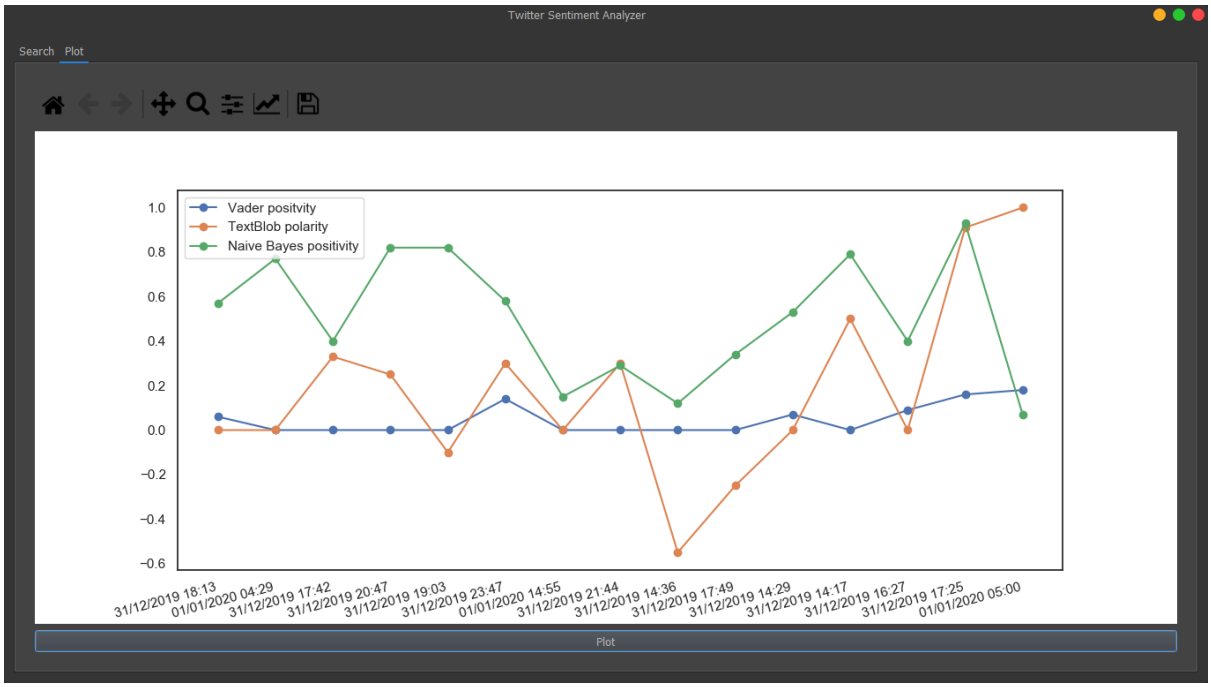

Fig. 4.4 – **fenêtre de graphe**

Chapitre 5

Conclusion

Dans ce projet, nous devions concevoir et développer une application desktop d'analyse des sentiments des tweets en temps réel.

Dans ce rapport, nous avons commencé par une présentation du contexte générale du projet, comme on a examiné l'étude théorique effectuée lors de la réalisation en abordant l'aspect conceptuel du projet, on a aussi discuté les différents choix techniques qui nous ont aidé à l'élaboration de notre application, avant de détailler les différentes étapes de l'implémentation du projet.

La démarche utilisée pour résoudre ce problème a tout de même quelques limites qui malheureusement demande plus de temps et de moyen afin de les aborder, on peut citer comme exemple le nombre des tweets fournis par TwitterAPI, qui se limite à x tweet, ainsi les humains peuvent exprimer leurs émotions de manière déguisée sans utiliser de vocabulaire émotif. Par exemple, "Mon travail m'oblige à travailler 25 heures par jour". Bien sûr, dans cette phrase, le locuteur exprime un sentiment négatif hyperbolique, suggérant qu'il doit travailler trop pour son travail. De telles expressions ne peuvent être saisies qu'à l'aide d'une base de connaissances du monde réel. En plus, la complexité des langues humains (anglaise dans notre cas) nous empêche à détecter très précisément les sentiments des tweets étudiés.

Dans les versions futures, Nous pourrions encore améliorer notre application en essayant d'extraire plus de fonctionnalités des tweets, en essayant différents types de fonctionnalités, en ajustant les paramètres du classificateur naïf de Bayes, ou en essayant un autre classificateur tous ensemble.

Bibliographie

[Bird et al., 2009] Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python : analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

[Bonner, 2019] Bonner, A. (2019). Getting started with google colab. https://towardsdatascience.com/.

[Goldberg, 2017] Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1) :1–309.

[Ho, 1995] Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

[Hu and Liu, 2004] Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

[Hutto and Gilbert, 2014] Hutto, C. J. and Gilbert, E. (2014). Vader : A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

[Klein and Manning, 2002] Klein, D. and Manning, C. D. (2002). Natural language grammar induction using a constituent-context model. In *Advances in neural information processing systems*, pages 35–42.

[Loria, 2018] Loria, S. (2018). textblob documentation. Technical report, Technical report.

[Luashchuk, 2019] Luashchuk, A. (2019). Why i think python is perfect for machine learning and artificial intelligence. https://towardsdatascience.com/.

[Martinez-Arroyo and Sucar, 2006] Martinez-Arroyo, M. and Sucar, L. E. (2006). Learning an optimal naive bayes classifier. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 1236–1239. IEEE.

[Murty and Devi, 2011] Murty, M. N. and Devi, V. S. (2011). *Pattern recognition : An algorithmic approach*. Springer Science & Business Media.

[Python-Twitter, 2016] Python-Twitter (2016). https://python-twitter.readthedocs.io/.

[Taboada et al., 2011] Taboada, M., Brooke, J., Tofiloski, M., Voll, K., and Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2) :267–307.

[TwitterAPI, 2019] TwitterAPI (2019). https://developer.twitter.com/en/docs.

[Vinet and Zhedanov, 2011] Vinet, L. and Zhedanov, A. (2011). A 'missing'family of classical orthogonal polynomials. *Journal of Physics A : Mathematical and Theoretical*, 44(8) :085201.

[Wahid et al., 2017] Wahid, H., Ahmad, S., Nor, M. A. M., and Rashid, M. A. (2017). Prestasi kecekapan pengurusan kewangan dan agihan zakat : perbandingan antara majlis agama islam negeri di malaysia. *Jurnal Ekonomi Malaysia*, 51(2) :39–54.

[Yi and Tian, 2011] Yi, C. and Tian, Y. (2011). Assistive text reading from complex background for blind persons. In *International workshop on camera-based document analysis and recognition*, pages 15–28. Springer.

Annexe A

Formation de notre modèle Naive Bayes pour l'analyse des sentiments sur Twitter

[20]:
```
import pandas as pd
import numpy as np
import nltk
nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

[20]:
```
True
```

[21]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[22]:
```
tweets = pd.read_csv("/content/drive/My Drive/Colab Notebooks/Tweets/tweets.
↪csv", header = None, names=["score", "text"]).drop(0)
tweets.head()
```

```
/usr/local/lib/python3.6/dist-packages/IPython/core/interactiveshell.py:2718:
DtypeWarning: Columns (0) have mixed types. Specify dtype option on import or
set low_memory=False.
interactivity=interactivity, compiler=compiler, result=result)
```

[22]:

	score	text
1	0	@switchfoot http://twitpic.com/2y1zl - Awww, t...
2	0	is upset that he can't update his Facebook by ...
3	0	@Kenichan I dived many times for the ball. Man...
4	0	my whole body feels itchy and like its on fire
5	0	@nationwideclass no, it's not behaving at all...

[0]:
```
from sklearn.utils import shuffle

tweets = shuffle(tweets)
```

[0]:
```
import re
from nltk.tokenize import word_tokenize
from string import punctuation
from nltk.corpus import stopwords

class PreProcessTweets:
def __init__(self):
self._stopwords = set(stopwords.words('english') + list(punctuation) +␣
↪['AT_USER', 'URL'])

def process_tweets(self, list_of_tweets):
processed_tweets = []
for index, tweet in list_of_tweets.iterrows():
processed_tweets.append((self._process_tweet(tweet["text"]),␣
↪tweet["score"]))
return processed_tweets

def _process_tweet(self, tweet):
tweet = tweet.lower()  # convert text to lower-case
tweet = re.sub(r'((www\.[^\s]+)|(https?://[^\s]+))', 'URL', tweet)  #␣
↪remove URLs
tweet = re.sub(r'@[^\s]+', 'AT_USER', tweet)  # remove usernames
tweet = re.sub(r'#([^\s]+)', r'\1', tweet)  # remove the # in #hashtag
tweet = word_tokenize(tweet)  # remove repeated characters
return [word for word in tweet if word not in self._stopwords]
```

d

[0]:
```
processor = PreProcessTweets()
processed_tweets = processor.process_tweets(tweets.head(50000))
```

[0]:
```
index = int(np.ceil(len(processed_tweets) * 0.8))

train = processed_tweets[:index]
test = processed_tweets[index:]
```

[0]:
```
def build_vocabulary(preprocessed_training_data):
all_words = []

for (words, _) in preprocessed_training_data:
all_words.extend(words)

word_list = nltk.FreqDist(all_words)
word_features = word_list.keys()

return word_features
```

[0]:
```
class NaiveBayesModel:
def __init__(self, preprocessed_set):
self.word_features = None
self.preprocessed_set = preprocessed_set
self.classifier = None
```

```
        def save(self):
        with open('/content/drive/My Drive/Colab Notebooks/Tweets/
↪my_naivebayes_classifier.pickle', 'wb') as file:
        pickle.dump(self.classifier, file)
        print("Model saved.")

        def load(self):
        with open('/content/drive/My Drive/Colab Notebooks/Tweets/
↪my_naivebayes_classifier.pickle', 'rb') as file:
        self.classifier = pickle.load(file)
        print("Model loaded.")

        def extract_features(self, tweet):
        tweet_words = set(tweet)
        features = {}
        for word in self.word_features:
        features["{}".format(word)] = (word in tweet_words)
        return features

        def result_labels(self, test_set):
        if self.classifier is not None:
        labels = [self.classifier.classify(self.extract_features(tweet[0])) for␣
↪tweet in test_set]
        return 100 * labels.count(1) / len(labels)

        def test_and_print_results(self, test_set):
        lb = 0
        for text, score in test_set:
        if score == 1:
        lb += 1
        real_score = 100 * lb / len(test_set)
        predicted_score = self.result_labels(test_set)

        print("Real score: {}".format(real_score))
        print("Predicted score: {}".format(predicted_score))

        def train_naive_bayes(self):
        self.word_features = build_vocabulary(self.preprocessed_set)
        training_features = nltk.classify.apply_features(self.extract_features,␣
↪self.preprocessed_set)
        self.classifier = nltk.NaiveBayesClassifier.train(training_features)
```

[0]:
```
        nb = NaiveBayesModel(train)
```

[0]:
```
        nb.train_naive_bayes()
```

[31]:
```
        nb.test_and_print_results(test)
```

```
Real score: 48.9
Predicted score: 52.3
```

[32]:
```
        import pickle
        nb.save()
```

```
Model saved.
```

[0]:
```
import csv
def save_vocabulary(file_path, word_list):
with open(file_path, 'w', newline='') as file:
writer = csv.writer(file)
for word in word_list:
writer.writerow([str(word)])
print("Dictionary saved.")
```

[34]:
```
save_vocabulary("/content/drive/My Drive/Colab Notebooks/Tweets/
↪word_features.csv", nb.word_features)
```

```
Dictionary saved.
```