



UFR de Sciences Appliquées et de Technologie

Mention : Informatique

Niveau : Master 2 GDIL

EC : Text Mining

UE : Extraction de connaissances

---

**Projet 5 : Fouille d'opinions (Opinion Mining)**

---

**Présenté par :**

- Faback Dieng (P26 830)
- Cheikhna Diaw (P31 5381)
- El hadji Leye (P27 1120)

**Sous la direction :**

✓ Pr Cheikh Bamba Dione

## PLAN

I. Introduction-----	4
II. Méthodes -----	5
1) Collection et Description des données -----	5
2) Analyses théoriques des algorithmes utilisés -----	8
3) Présentation technique (Implémentation)-	12
III.Evaluation -----	22
1) Présentation et discussions des résultats ---	22
2) Description des expériences -----	26
IV.Conclusion -----	27
V. Références -----	28

## Liste des Figures

Figure 1: Informations de la base de données.....	5
Figure 2: Informations de la base de données avec les deux colonnes (text et airline_sentiment).6	6
Figure 3: Nombre d'occurrences des tweets.....	7
Figure 4: Nombre d'occurrences des tweets après suppression des tweets neutres .....	7
Figure 5: Comparaison réseau de neurones artificiel et neurone humain .....	10
Figure 6: Diagramme de performance entre le machine learning et le deep learning .....	12
Figure 7: Fonction de nettoyage des tweets .....	14
Figure 8: Remplacement des valeurs de la variable airline_sentiment en numérique .....	14
Figure 9: Résultats après changement des valeurs en numérique .....	15
Figure 10: Elimination des tweets neutres .....	15
Figure 11: Résultats après élimination des tweets neutres .....	16
Figure 12: Remplacement des valeurs négatives par 0 .....	16
Figure 13: Résultats après remplacement des valeurs négatives par 0 .....	16
Figure 14: Division des données .....	17
Figure 15: Implémentation du modèle SVM.....	18
Figure 16: Implémentation du modèle Régression Logistique .....	18
Figure 17: Implémentation du modèle Arbre de décision.....	19
Figure 18: Implémentation du modèle KNN.....	19
Figure 19: Importation des bibliothèques pour le deep learning.....	20
Figure 20: Implémentation du modèle monocouche .....	20
Figure 21: Implémentation du modèle multicouches.....	21
Figure 22: Résultats du modèle SVM .....	22
Figure 23: Résultats du modèle Régression Logistique.....	23
Figure 24: Résultats du modèle Arbre de décision .....	23
Figure 25: Résultats du modèle KNN .....	24
Figure 26: Résultats du modèle monocouche.....	24
Figure 27: Résultats du modèle multicouches .....	25

# I.Introduction

Dans notre société actuelle, exprimer nos opinions est devenu un acte quotidien et incontournable. Que ce soit en ligne, dans les médias sociaux ou dans notre entourage, les opinions abondent, façonnant ainsi notre perception du monde.

C'est pourquoi on s'est posé la question à savoir dans quelle mesure la fouille d'opinions peut-elle être utilisée de manière fiable et précise pour comprendre les opinions des populations et les utiliser dans la prise de décision ?

Les objectifs sont de comprendre les enjeux de la fouille d'opinions, d'évaluer la fiabilité et la précision des résultats obtenus, d'analyser les implications éthiques et de confidentialité, et de proposer des mesures d'amélioration et de réglementation pour maximiser les avantages de la fouille d'opinions tout en minimisant les risques.

Nous sommes intéressés par ce sujet car nous croyons fermement en l'importance de comprendre les opinions et les attitudes des individus dans notre société en constante évolution. En analysant les données, nous espérons découvrir de nouvelles perspectives et des tendances émergentes qui pourraient avoir un impact sur les entreprises, les décideurs politiques et la société dans son ensemble.

Notre étude se concentre sur les méthodes utilisées dans la fouille d'opinions et la manière dont elles sont mises en œuvre pour obtenir des résultats fiables. Notre objectif principal est de comprendre comment les données sont collectées, quelles sont les différentes approches théoriques des algorithmes utilisés et comment ces méthodes sont techniquement mises en œuvre.

Le premier volet de notre étude se concentre sur la collecte et la description des données, ensuite nous nous pencherons sur l'analyse théorique des algorithmes utilisés et en fin sur la présentation technique de l'implémentation de ces méthodes.

Dans la deuxième partie de notre étude, nous évaluerons les résultats obtenus à partir des méthodes présentées précédemment. Nous présenterons et discuterons des résultats. Et en fin, nous évaluerons également la robustesse et la fiabilité des résultats obtenus.

## II.Méthodes

### 1) Collection et Description des données

Comme méthode de recherche nous avons consulté des livres, des articles, des rapports académiques et des études de cas pour recueillir des informations pertinentes sur le sujet. Nous mettrons en évidence les caractéristiques du SVM, y compris son principe de base, sa capacité de classification binaire et multi classe, la fonction de décision, la sélection du noyau.

Les données à étudier ont été recueillies sur le site de kaggle, elles sont de type csv. Elles ont été collectées à partir d'un travail d'analyse de sentiments sur les problèmes de chaque grande compagnie aérienne américaine.

```
data = pd.read_csv("Tweets.csv", encoding="UTF-8")
data
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN
...	...	...	...	...	...	...	...

*Figure 1: Informations de la base de données*

Nous avons extrait les colonnes pertinentes à savoir le texte du tweet et le sentiment associé.

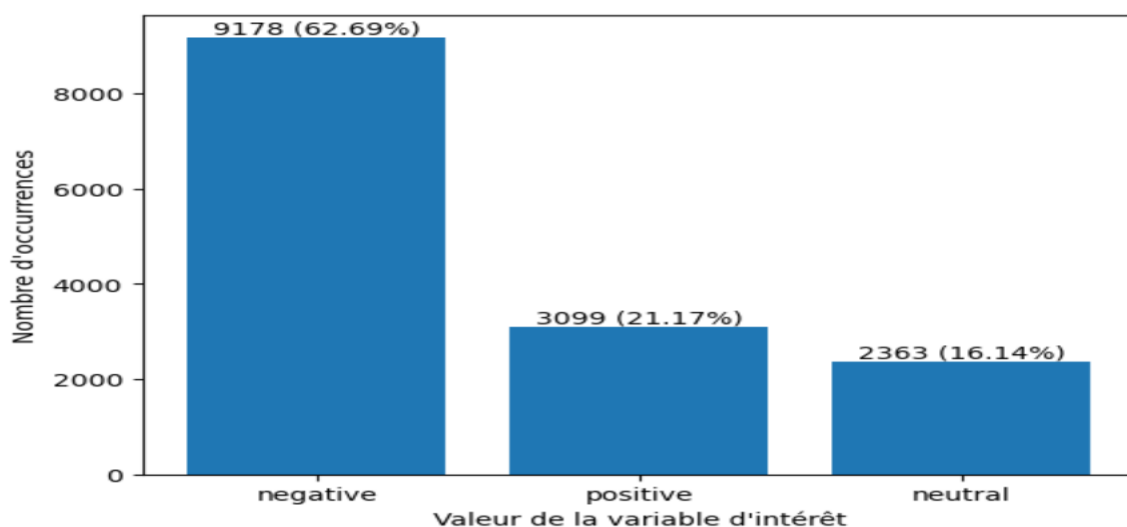
```
data = data[['text', 'airline_sentiment']]
data
```

	text	airline_sentiment
0	@VirginAmerica What @dhepburn said.	neutral
1	@VirginAmerica plus you've added commercials t...	positive
2	@VirginAmerica I didn't today... Must mean I n...	neutral
3	@VirginAmerica it's really aggressive to blast...	negative
4	@VirginAmerica and it's a really big bad thing...	negative
...	...	...
14635	@AmericanAir thank you we got on a different f...	positive
14636	@AmericanAir leaving over 20 minutes Late Flig...	negative
14637	@AmericanAir Please bring American Airlines to...	neutral
14638	@AmericanAir you have my money, you change my ...	negative
14639	@AmericanAir we have 8 ppl so we need 2 know h...	neutral

14640 rows × 2 columns

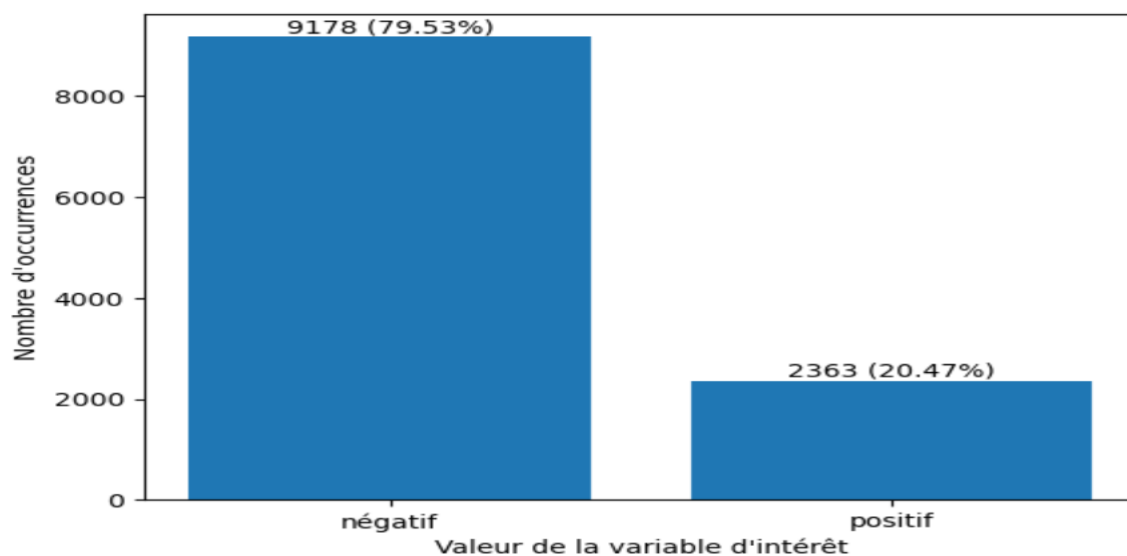
*Figure 2: Informations de la base de données avec les deux colonnes (text et airline\_sentiment)*

Avant d'utiliser ces données collectées nous effectuerons un travail de nettoyage pour éliminer les éléments indésirables tels que les balises HTML, les symboles de ponctuation, les URL, les caractères spéciaux, etc. L'objectif de ce nettoyage est de préparer les données afin qu'elles soient utilisables par la suite et de garantir la qualité et la cohérence des résultats de l'analyse. Pour évaluer les performances du modèle, nous diviserons les données collectées en ensemble d'apprentissage et de test. La répartition dépendra de plusieurs facteurs tels que la taille des données, la disponibilité des données et les bonnes pratiques de la communauté scientifique. Nous pouvons utiliser une répartition de 80% pour l'apprentissage et 20% pour les tests.



*Figure 3: Nombre d'occurrences des tweets*

Le graphe ci-dessus nous fait part du nombre d'occurrences (positif, négatif et neutre) des tweets collectés et montre un déséquilibre entre les trois classes (positive, négative et neutre). Ce déséquilibre peut en effet représenter un problème dans le contexte où le modèle apprend plus sur les tweets négatifs. Cela rendra notre modèle très efficace pour classer correctement les tweets négatifs mais en contrepartie le manque de tweets positifs et neutres impactera négativement les performances du modèle quand il tentera de classer ces derniers. Par la suite nous avons jugé nécessaire de supprimer les tweets neutres car ils ne sont pas pertinents pour notre étude.



*Figure 4: Nombre d'occurrences des tweets après suppression des tweets neutres*

Le graphe ci-dessus nous montre du nombre d'occurrences (positif, négatif) après suppression des tweets neutres.

## 2) Analyses théoriques des algorithmes utilisés

Pour l'analyse théorique on va essayer de comparer les algorithmes utilisés par le machine learning et ceux utilisés par le deep learning.

### **Machine Learning**

Le machine Learning, ou apprentissage automatique, est un sous-ensemble de l'intelligence artificielle qui consiste à créer des modèles et des algorithmes permettant à une machine d'apprendre et de s'améliorer à partir de données, sans être explicitement programmé. Il permet à la machine de comprendre un problème et d'effectuer des prédictions ou de prendre des décisions en fonction des informations qu'elle a apprises. Concernant les algorithmes de machine learning, nous allons en explorer quelques-uns.

#### *Machine à vecteur de support (SVM)*

Notre choix d'algorithme s'est porté sur SVM car les SVM offre généralement une excellente précision de prédiction, en particulier dans des contextes où la séparation entre les classes est claire. Ils utilisent des noyaux (kernel) qui permettent de projeter les données d'entrée dans des espaces de dimensions supérieures, ce qui leur permet de séparer les données non linéairement séparables. De même, ils sont relativement robustes face aux données bruitées, grâce à leur capacité à ignorer les observations extrêmes ou atypiques. L'entraînement d'un SVM consiste à résoudre un problème d'optimisation convexe, ce qui garantit une solution globale et évite les minimas locaux.

#### *Régression Logistique*

La régression logistique est une technique d'apprentissage automatique supervisé utilisé pour la classification binaire. Elle est principalement utilisée lorsque la variable cible (ou variable dépendante) est une variable binaire (par exemple, oui/non) et que l'on souhaite prédire la probabilité d'appartenance à une classe spécifique. La régression logistique utilise une fonction logistique pour modéliser la probabilité d'appartenance à la classe positive en fonction des variables indépendantes (ou variables explicatives). La fonction logistique transforme la somme pondérée des variables indépendantes en une valeur



comprise entre 0 et 1, qui peut être interprétée comme la probabilité d'appartenance à la classe positive.

Le modèle de régression logistique est estimé à l'aide de la méthode du maximum de vraisemblance, ou les poids (ou coefficients) des variables indépendantes sont ajustés pour maximiser la probabilité d'observer les données réelles étant donné le modèle.

La régression logistique est largement utilisée dans divers domaines, tels que la médecine, l'économie, le marketing, ou la classification binaire est fréquemment utilisée pour prendre des décisions ou prédire des événements.

### *Arbre de décision*

Un arbre de décision est une technique d'apprentissage automatique supervisée utilisée pour la classification et la régression. Il représente un modèle prédictif sous la forme d'une structure en arborescence où chaque nœud interne représente une décision basée sur une caractéristique particulière, chaque branche correspond à une valeur possible de cette caractéristique, et chaque feuille représente la prédiction finale.

L'arbre de décision est construit en sélectionnant de manière récursive les caractéristiques qui divisent le mieux l'ensemble de données en sous-ensembles homogènes en terme de classe (dans le cas de la classification) ou de prédiction (dans le cas de la régression). Cette division est basée sur des mesures d'*impureté* ou de *gain d'information*, telles que l'*entropie*.

Lorsque l'arbre de décision est construit, il peut être utilisé pour effectuer des prédictions sur de nouvelles observations. Pour cela, il suit le chemin de haut en bas, en fonction des valeurs des caractéristiques, jusqu'à atteindre une feuille qui fournit la prédiction finale.

### *KNN*

K-plus proches voisins est un algorithme d'apprentissage automatique qui est utilisé pour la classification et la régression. Il est basé sur le principe selon lequel les observations similaires tendent à se regrouper dans l'espace des caractéristiques.

Dans le cas de la classification KNN, lorsqu'on lui présente une nouvelle observation dont la classe est inconnue, l'algorithme KNN examine les k observations les plus proches dans l'espace des caractéristiques (mesurées à travers des attributs ou des variables indépendantes) à partir de l'ensemble de données d'entraînement. Il utilise

ensuite la classe majoritaire de ces  $k$  voisins pour prédire la classe de la nouvelle observation.

La *mesure de similarité* utilisée pour déterminer les voisins les plus proches peut être une *distance euclidienne*, une *distance de Manhattan* ou une autre mesure de similarité adaptée aux types de données sur lesquelles l'algorithme est appliqué.

Dans le cas de la régression KNN, au lieu de prédire une classe, l'algorithme prédit une valeur numérique en utilisant la *moyenne* ou la *médiane* des valeurs des  $k$  voisins les plus proches.

## Deep Learning

Le deep learning, ou apprentissage profond en français, est une branche de l'intelligence artificielle qui se concentre sur l'apprentissage automatique et l'analyse des données à partir de réseaux de neurones artificiels profonds.

Le concept clé du deep learning est l'utilisation de réseaux de neurones artificiels profonds. Ces réseaux sont composés de plusieurs couches de neurones interconnectés, qui sont conçus pour simuler le fonctionnement du cerveau humain et traiter des données de manière hiérarchique.

### Les réseaux de neurones artificiels

> L'objectif du réseau neuronal est de résoudre des problèmes de la même manière que le cerveau humain

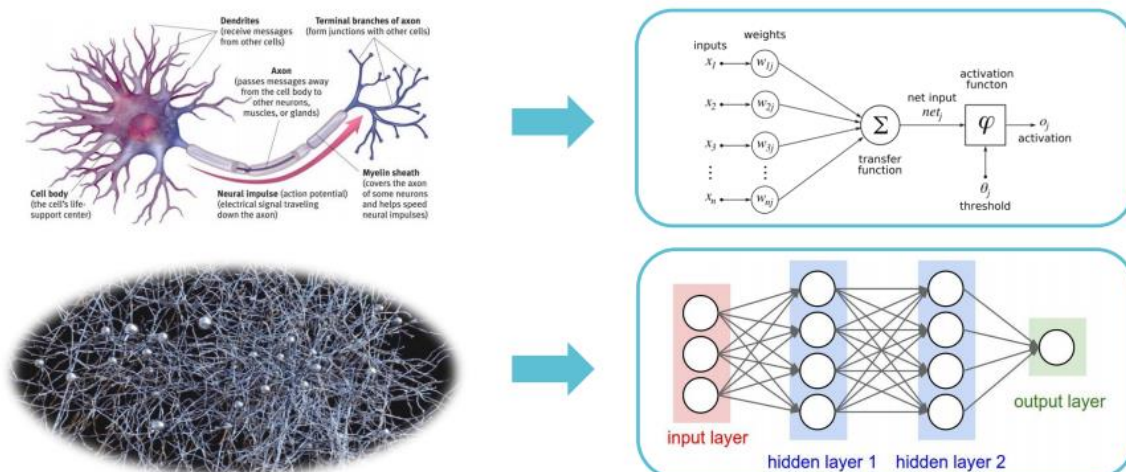


Figure 5: Comparaison réseau de neurones artificiel et neurone humain

Contrairement aux réseaux de neurones traditionnels, les réseaux de neurones profonds ont un grand nombre de couches et de paramètres, ce qui leur donne une capacité accrue à capturer les caractéristiques subtiles et complexes des données. Cette profondeur supplémentaire permet aux réseaux de neurones de construire des niveaux d'abstraction progressifs, ce qui permet d'extraire des caractéristiques de plus haut niveau à partir des caractéristiques de bas niveau.

Le deep Learning a révolutionné de nombreux domaines tels que la vision par ordinateur, la reconnaissance vocale pour ne citer que cela.

### *Réseau de neurones monocouche*

Un réseau de neurone monocouche, également appelé perceptron monocouche, est un type simple de réseau de neurones artificiels composé d'une seule couche de neurones. Chaque neurone est connecté à toutes les entrées du réseau et possède une seule sortie.

### *Réseau de neurones multicouches*

Un réseau de neurones multicouches est un type de réseau de neurones artificiels qui comporte plusieurs couches de neurones, chacune étant connectée à la couche suivante. Il est également connu sous le nom de réseau de neurones profonds en raison de sa profondeur qui peut varier en fonction du nombre de couches cachées.

## **Apprentissage profond vs Apprentissage automatique**

L'apprentissage profond implique l'application de réseaux de neurones profonds avec plus de couches et plus de données que les algorithmes d'apprentissage automatique traditionnels.

Ceci est également utile, car même si la quantité de données augmente, les performances des algorithmes d'apprentissage automatique traditionnels ne peuvent pas être améliorées après un certain temps tandis que les performances des algorithmes d'apprentissage profond sont proportionnelles à la quantité de données.

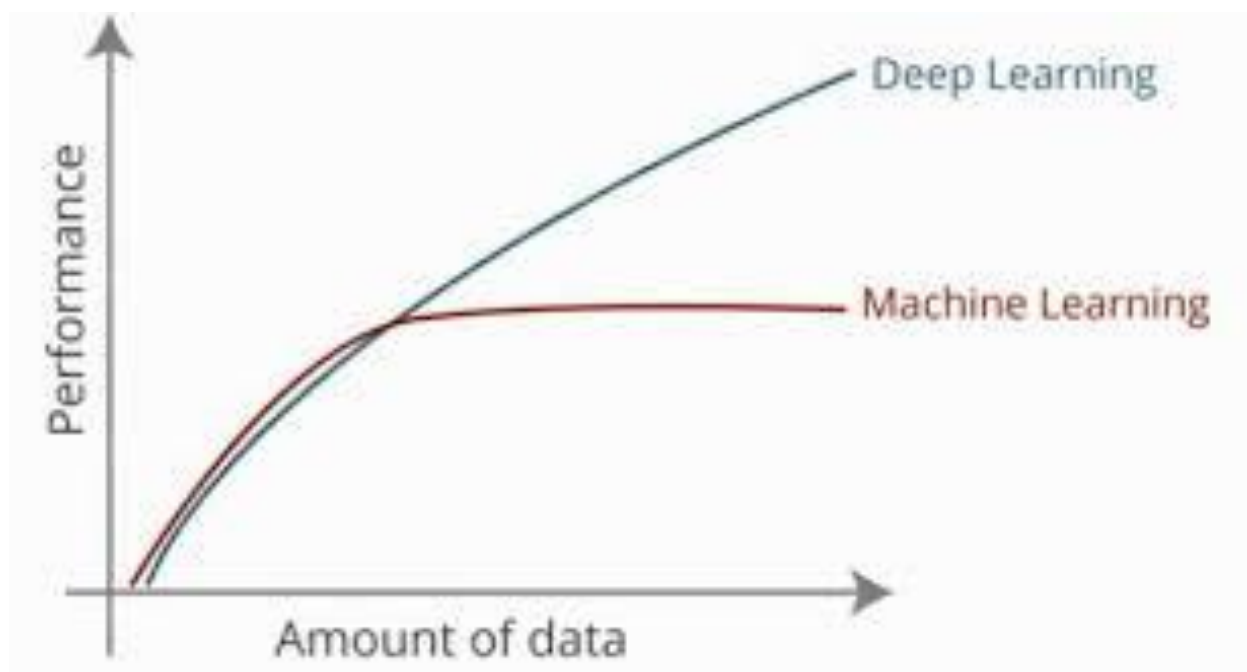


Figure 6: Diagramme de performance entre le machine learning et le deep learning

### 3) Présentation technique (Implémentation)

Le prétraitement des tweets est une étape essentielle dans le domaine du traitement du langage naturel (NLP) pour améliorer la qualité et l'efficacité de l'analyse des tweets.

Dans cette présentation technique, nous explorerons les différentes étapes de prétraitement des tweets et l'implémentation des algorithmes choisis.

#### *Suppression des caractères spéciaux et des ponctuations*

Dans cette première étape, nous éliminons les caractères spéciaux et les ponctuations présents dans les tweets. Cela inclut des symboles tels que les @, #, \$, %, etc., ainsi que les ponctuations telles que les points, les virgules, etc. Cette étape vise à éliminer le bruit inutile du texte et à simplifier les données.

#### *Conversion du texte en minuscule*

La conversion du texte en minuscule est une étape courante dans le prétraitement des données textuelles. Cela permet d'uniformiser le texte et de réduire les variations dues à la classe (majuscules/minuscules). Cela facilite également la comparaison et l'analyse ultérieure, car les mots en minuscules sont considérés comme identiques,

indépendamment de leur classe.

### *Retrait des liens web*

Les tweets peuvent contenir des liens web, tels que des url ou des adresses de site web. Dans cette étape, nous supprimons ces liens car ils ne nous fournissent aucune information supplémentaire pour notre étude. Cela permet également de réduire le bruit dans les données et de rendre le texte plus lisible et compacte.

### *Retrait des nombres*

Dans cette étape, nous retirons les nombres présents dans les tweets. Les chiffres ne contribuent généralement pas à l'analyse du sentiment ou à la compréhension du texte, à moins qu'ils ne soient spécifiquement pertinents pour l'objectif du projet. En retirant les nombres, nous réduisons également le bruit et simplifions le texte.

### *Tokenisation du texte*

La tokénisation est une étape cruciale dans le prétraitement des tweets. Elle consiste à diviser le texte en mots ou « tokens » individuels. Les mots ou les tokens facilitent l'analyse et le traitement ultérieur du texte. La tokénisation peut être réalisée en utilisant des méthodes telles que la séparation par des espaces ou l'utilisation d'algorithmes plus avancés comme le tokeniseur NLTK.

### *Suppression des mots vides*

Dans cette étape, nous supprimons les mots vides (stop Word). Les mots vides sont des mots très commun dans la langue (par exemple : et, de, la, les, etc.) qui n'apportent généralement pas de valeur informative dans l'analyse des tweets. En les retirant, nous réduisons la dimensionnalité des données et nous nous concentrons sur les mots plus significatifs.

### *Lemmatisation*

La lemmatisation est une étape optionnelle mais souvent utilisée dans le prétraitement des tweets. Elle consiste à réduire les mots à leur forme de base ou canonique, appelé lemme. Par exemple, les mots « courir », « courais », « courait » seraient tous

lemmatisés en « courir ». La lemmatisation aide à normaliser le texte et à réduire le bruit en regroupant les variantes d'un même mot.

```
# Fonction de nettoyage
def clean_text(text):
    # Supprimer des caractères spéciaux et de la ponctuation
    text = re.sub(r"^[^\w\s]", "", text)
    # Conversion en minuscule
    text = text.lower()
    # retrait des liens web (http et https)
    text = re.sub(r'http\S+', '', text)
    # retrait des nombres
    text = re.sub("[0-9_]", "", text)
    # Tokénisation du texte
    tokens = word_tokenize(text)
    # Suppression des mots vides
    stop_words = set(stopwords.words("english"))
    tokens = [word for word in tokens if word not in stop_words]
    tokens # Affichage des tokens

    # Lemmatisation
    lem=WordNetLemmatizer()
    lemmatisation = [lem.lemmatize(mot) for mot in tokens]

    # Joindre Les tokens en une seule chaîne
    cleaned_text = "".join(mot for mot in lemmatisation)

    return cleaned_text

# Appliquer Le nettoyage sur La colonne texte
data['text'] = data['text'].apply(clean_text)
data
```

Figure 7: Fonction de nettoyage des tweets

## Remplacement des valeurs en numérique de la variable airline\_sentiment

```
# -1 si Le text analyse d'opinion : negative (negative)
# 0 si Le text analyse d'opinion : neutral (neutre)
# 1 si Le text analyse d'opinion : positive (positive)

class_map={'negative':-1, 'neutral':0, 'positive':1}
print(class_map['negative'])
print(class_map['neutral'])
print(class_map['positive'])

-1
0
1
```

Figure 8: Remplacement des valeurs de la variable airline\_sentiment en numérique

L'intérêt de ce code est de pouvoir convertir facilement des étiquettes d'analyse d'opinions en valeurs numériques correspondantes. Cela peut être utile lors de la préparation des données pour l'entraînement du modèle de machine learning ou lors de l'évaluation des résultats d'un modèle. Par exemple, si nous avons un ensemble de données comprenant des étiquettes d'analyse d'opinions, nous pouvons les convertir en

utilisant ce dictionnaire pour les représenter sous forme numérique, ce qui facilite l'entraînement d'un modèle de classification binaire.

	text	airline_sentiment
0	virginamericadhepburnsaid	0
1	virginamericaplusyouveaddedcommercialexperienc...	1
2	virginamericadidnttodaymustmeanneedtakeanother...	0
3	virginamericareallyaggressiveblastobnoxiousent...	-1
4	virginamericareallybigbadthing	-1
...	...	...
14635	americanairthankgotdifferentflightchicago	1
14636	americanairleavingminutelateflightwarningcommu...	-1
14637	americanairpleasebringamericanairlineblackberry	0
14638	americanairmoneychangeflightdontanswerphonesug...	-1
14639	americanairpplneedknowmanyseatnextflightplzput...	0

14640 rows × 2 columns

*Figure 9: Résultats après changement des valeurs en numérique*

```
data = data[data['airline_sentiment'] != 0]
data
```

*Figure 10: Elimination des tweets neutres*



	text	airline_sentiment
1	virginamericaplusyouveaddedcommercialexperienc...	1
3	virginamericareallyaggressiveblastobnoxiousent...	-1
4	virginamericareallybigbadthing	-1
5	virginamericaseriouslywouldpayflightseatdidntp...	-1
6	virginamericayesnearlyeverytimeflyvxearwormwon...	1
...	...	...
14633	americanairflightcancelledflightledleavingtomo...	-1
14634	americanairrightcuedelaysokhand	-1
14635	americanairthankgotdifferentflightchicago	1
14636	americanairleavingminutelateflightwarningcommu...	-1
14638	americanairmoneychangeflightdontanswerphonesug...	-1

11541 rows × 2 columns

Figure 11: Résultats après élimination des tweets neutres

```
data['airline_sentiment'] = data['airline_sentiment'].replace(-1, 0)
data
```

Figure 12: Remplacement des valeurs négatives par 0

	text	airline_sentiment
1	virginamericaplusyouveaddedcommercialexperienc...	1
3	virginamericareallyaggressiveblastobnoxiousent...	0
4	virginamericareallybigbadthing	0
5	virginamericaseriouslywouldpayflightseatdidntp...	0
6	virginamericayesnearlyeverytimeflyvxearwormwon...	1
...	...	...
14633	americanairflightcancelledflightledleavingtomo...	0
14634	americanairrightcuedelaysokhand	0
14635	americanairthankgotdifferentflightchicago	1
14636	americanairleavingminutelateflightwarningcommu...	0
14638	americanairmoneychangeflightdontanswerphonesug...	0

11541 rows × 2 columns

Figure 13: Résultats après remplacement des valeurs négatives par 0

### Division des données en ensemble d'entraînement et de test

La division des données en ensemble d'entraînement et de test est une étape essentielle dans le processus de développement de modèles d'apprentissage automatique. Elle consiste à séparer l'ensemble de données initial en deux ensembles distincts :



l'ensemble d'entraînement et l'ensemble de test. L'ensemble d'entraînement est utilisé pour former (ou ajuster) le modèle, tandis que l'ensemble de test est utilisé pour évaluer les performances du modèle de manière réaliste. Elle permet également de détecter des problèmes tels que le sur apprentissage (overfitting), ou le modèle mémorise les données d'entraînement mais ne parvient pas à généraliser correctement. Il est recommandé d'allouer une plus grande proportion de données à l'ensemble d'entraînement (par exemple 70-80%) et une plus petite portion à l'ensemble test (par exemple 20-30%). Cependant, ces proportions peuvent varier en fonction de la taille totale de l'ensemble de données et de la complexité du problème.

## Machine Learning

### Modélisation

#### Division des données en ensemble d'entraînement et de test

```
: ▶ X_train, X_test, y_train, y_test = train_test_split(data['text'], data['airline_sentiment'], test_size=0.2,
: ▶ data.shape
6]: (11541, 2)
```

*Figure 14: Division des données*

## Machine à vecteur de Support (SVM)

```
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Définir Le modèle SVM avec un TfidfVectorizer
svm_model = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('svm', SVC(kernel='linear', C=1, random_state=0))
])

# Entraîner Le modèle SVM
svm_model.fit(X_train, y_train)

# Prédiction sur l'ensemble de test
y_pred = svm_model.predict(X_test)

# Évaluation du modèle
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy:f}")
print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Classification Report:\n{class_report}")
```

Figure 15: Implémentation du modèle SVM

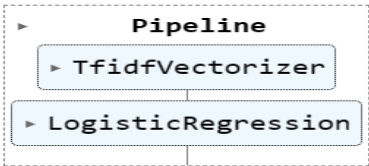
## Régression Logistique

### Comparaison avec les autres algorithmes

#### 1) Logistic Regression

```
► # Définir Le modèle de RÉGRESSION LOGISTIQUE avec un TfidfVectorizer
lr_model = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('lr', LogisticRegression(solver='liblinear', random_state=0))
])
# Construction du Modèle
lr_model.fit(X_train, y_train)
```

9]:



```
► # Calcul du taux de performance du Modèle
y_pred = lr_model.predict(X_test)
accuracy = lr_model.score(X_test, y_test)
print('Accuracy Logistic Regression Test: ', accuracy)
```

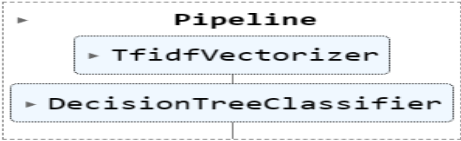
Figure 16: Implémentation du modèle Régression Logistique

## 2) Arbre de décision

```
: ▶ # Importation des bibliothèques
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier

: ▶ # Définir Le modèle d'ARBRE DE DECISION avec un TfidfVectorizer
ad_model = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('ad', tree.DecisionTreeClassifier())
])
# Construction d'un arbre de décision
ad_model.fit(X_train, y_train)

33]:
```



```
: ▶ # Calcul du taux de performance
accAD = ad_model.score(X_test, y_test)
print('Accuracy Arbre Decision Test: ', accAD)
```

The diagram shows a 'Pipeline' box containing two sub-boxes: 'TfidfVectorizer' and 'DecisionTreeClassifier'. They are connected by a vertical line, indicating a sequential process.

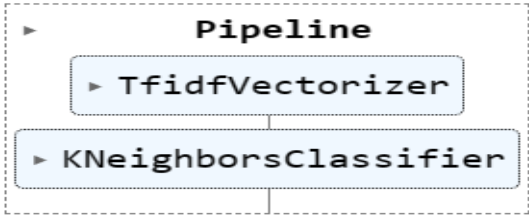
Figure 17: Implémentation du modèle Arbre de décision

## 3) KNN

```
▶ # Importation des bibliothèques
from sklearn.neighbors import KNeighborsClassifier

▶ # Définir Le modèle KNN avec un TfidfVectorizer
knn_model = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('knn', KNeighborsClassifier(n_neighbors=1))
])
# Construction du modèle KNN
knn_model.fit(X_train, y_train)

7]:
```



```
▶ # Calcul du taux de performance
score = knn_model.score(X_test, y_test)
print('Accuracy KNN Test: ', score)
```

The diagram shows a 'Pipeline' box containing two sub-boxes: 'TfidfVectorizer' and 'KNeighborsClassifier'. They are connected by a vertical line, indicating a sequential process.

Figure 18: Implémentation du modèle KNN

En résumé pour tous les modèles ci-dessus on importe d'abord les bibliothèques nécessaires, ensuite on essaie de définir le modèle avec un TfidfVectorizer qui est un outil utilisé dans le traitement du langage naturel et de l'apprentissage automatique pour convertir du texte en une représentation numérique vectorielle, il est basé sur le concept de pondération de la fréquence inverse du document (TF-IDF), construire celui-ci. Et en fin, on calcule le taux de performance de ces modèles.

## Deep Learning

### Réseau de neurones (mono couche)

#### 1) Mono couche

```
# Chargement des packages
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, SpatialDropout1D
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

Figure 19: Importation des bibliothèques pour le deep learning

```
# Tokenisation
max_words = 5000
tokenizer = Tokenizer(num_words=max_words, split=' ')
tokenizer.fit_on_texts(X_train)
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)

# Remplissage des séquences
max_len = max(len(x) for x in X_train_seq)
X_train_pad = pad_sequences(X_train_seq, maxlen=max_len)
X_test_pad = pad_sequences(X_test_seq, maxlen=max_len)

# Création du modèle avec une seule couche
embedding_dim = 128
model = Sequential()
model.add(Embedding(max_words, embedding_dim, input_length=max_len))
model.add(SpatialDropout1D(0.2))
model.add(Dense(1, activation='sigmoid'))

# Compiler Le modèle
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Entraîner Le modèle
batch_size = 64
epochs = 5
model.fit(X_train_pad, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test_pad, y_test))
```

Figure 20: Implémentation du modèle monocouche

## 2) Multi couche

```
▶ # Tokenisation
max_words = 50000
tokenizer = Tokenizer(num_words=max_words, split=' ')
tokenizer.fit_on_texts(X_train)
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)

# Remplissage des séquences
max_len = max(len(x) for x in X_train_seq)
X_train_pad = pad_sequences(X_train_seq, maxlen=max_len)
X_test_pad = pad_sequences(X_test_seq, maxlen=max_len)

# Création du modèle
embedding_dim = 128
model = Sequential()
model.add(Embedding(max_words, embedding_dim, input_length=max_len))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))

# Compiler le modèle
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 21: Implémentation du modèle multicouches

En résumé pour les modèles monocouche et multicouches dans un premier temps, nous avons importés les packages nécessaires. Ensuite, les données textuelles sont tokenisées à l'aide de la classe Tokeniser et les séquences obtenues sont remplies pour qu'elles aient toutes la même longueur.

Ensuite, le réseau de neurones est créé en utilisant la classe Sequential de keras. Il comprend une couche d'embedding pour apprendre une représentation dense de mots, suivie d'une couche de drop-out spatial pour éviter le sur apprentissage. Une couche *LMST* (pour le réseau de neurone multicouches) est ajoutée pour capturer les dépendances à long terme dans les séquences. Enfin, une couche dense avec une fonction d'activation sigmoïde est ajoutée pour la classification binaire.

Le modèle est ensuite compilé en spécifiant la fonction de perte, l'optimiseur et la métrique à utiliser lors de l'entraînement. Le modèle est ensuite entraîné sur les données d'entraînement en spécifiant la taille du lot et le nombre d'epochs. Une fois l'entraînement terminé, l'exactitude du modèle est évaluée en utilisant les données de test et le résultat est affiché.

### III.Evaluation

#### 1) Présentation et discussions des résultats

Le modèle a été évalué en termes d'accuracy. Les résultats montrent une précision sur l'ensemble de test. Pour cela nous allons comparer la performance des différents algorithmes utilisés en machine learning et ceux utilisés en deep learning.

#### Machine Learning

##### *Machine à vecteur de support (SVM)*

```
Accuracy: 0.809008
Confusion Matrix:
[[1835    1]
 [ 440   33]]
Classification Report:
              precision    recall  f1-score   support

     0       0.81         1.00         0.89       1836
     1       0.97         0.07         0.13         473

 accuracy                   0.81       2309
 macro avg                   0.53       2309
 weighted avg                0.81       2309
```

Figure 22: Résultats du modèle SVM

Pour le SVM on a une précision globale (*accuracy*) de 0.809, ce qui indique que notre modèle prédit correctement la classe des tweets dans 80,9% des cas. Ensuite on a la *matrice de confusion* qui montre la performance de notre modèle en termes de *vrais positifs*, *vrais négatifs* et l'erreur de classification. Les résultats de la matrice de confusion montrent que 1835 tweets sont correctement classés comme négatifs (vrais positifs), 33 tweets sont correctement classés comme positifs (vrais négatifs), 1 tweet est incorrectement classé comme négatif (faux positifs) et 440 tweets sont incorrectement classés comme positifs (faux négatifs). En fin, on a le rapport de classification qui fournit des mesures plus détaillées pour chaque classe.

**Pour la classe 0 (négatif)** la précision est de 0.81, ce qui signifie que 81% des tweets classés négatifs le sont réellement. Le rappel (*recall*) est de 1, indiquant que notre modèle identifie correctement tous les tweets négatifs. Le *F1-score* est de 0.89, une mesure qui combine la précision et le rappel, ou la valeur de 1 est idéale. Le *support* est de 1836, ce qui correspond au nombre total de tweets négatifs dans notre ensemble de

données test.

**Pour la classe 1 (positif)** la précision est de 0.97, ce qui indique que 97% des tweets classés positifs le sont réellement. Le rappel est de 0.07, ce qui suggère que notre modèle a du mal à identifier les tweets positifs. Le F1-score est de 0.13, ce qui indique une faible performance pour cette classe, cela s'explique par le déséquilibre noté entre les deux classes. Le support est de 473, ce qui correspond au nombre total de tweets positifs dans notre ensemble de données test.

### Logistique Régression

	precision	recall	f1-score	support
0	0.80	1.00	0.89	1836
1	1.00	0.04	0.07	473
accuracy			0.80	2309
macro avg	0.90	0.52	0.48	2309
weighted avg	0.84	0.80	0.72	2309

Figure 23: Résultats du modèle Régression Logistique

### Arbre de décision

	precision	recall	f1-score	support
0	0.81	1.00	0.89	1836
1	0.97	0.07	0.13	473
accuracy			0.81	2309
macro avg	0.89	0.53	0.51	2309
weighted avg	0.84	0.81	0.74	2309

Figure 24: Résultats du modèle Arbre de décision

Pour l'arbre de décision, on note les mêmes performances que le machine à vecteur de support.

## KNN

	precision	recall	f1-score	support
0	0.95	0.02	0.04	1836
1	0.21	1.00	0.34	473
accuracy			0.22	2309
macro avg	0.58	0.51	0.19	2309
weighted avg	0.80	0.22	0.10	2309

Figure 25: Résultats du modèle KNN

Pour les résultats du KNN on constate qu'on a un taux de performance très faible (0.22) cela est dû au déséquilibre noté dans les tweets positifs et négatifs ce qui peut entrainer des biais dans les performances du modèle.

## Deep Learning

### Réseaux de neurones (mono couche)

```
Epoch 1/5
145/145 [=====] - 3s 11ms/step - loss: 0.6047 - accuracy: 0.7509 - val_loss: 0.4972 - val_accuracy: 0.8060
Epoch 2/5
145/145 [=====] - 1s 8ms/step - loss: 0.5318 - accuracy: 0.8614 - val_loss: 0.4930 - val_accuracy: 0.8077
Epoch 3/5
145/145 [=====] - 1s 8ms/step - loss: 0.4296 - accuracy: 0.9156 - val_loss: 0.4890 - val_accuracy: 0.8077
Epoch 4/5
145/145 [=====] - 1s 7ms/step - loss: 0.3187 - accuracy: 0.9161 - val_loss: 0.4894 - val_accuracy: 0.8077
Epoch 5/5
145/145 [=====] - 1s 8ms/step - loss: 0.2595 - accuracy: 0.9161 - val_loss: 0.4920 - val_accuracy: 0.8077
73/73 [=====] - 0s 3ms/step - loss: 0.4920 - accuracy: 0.8077
Accuracy: 80.77%
```

Figure 26: Résultats du modèle monocouche

D'après les résultats, nous pouvons observer un comportement positif au cours des différents epochs. L'erreur (Loss) diminue de manière significative, passant de 0.6047 à 0.2595, ce qui suggère que le réseau de neurone parvient à ajuster les poids pour réduire l'erreur de prédiction. De plus, l'accuracy augmente progressivement de 0.7509 à 0.9161, ce qui indique que le modèle améliore sa capacité à classer correctement les tweets.



## Réseaux de neurones (multi couche)

```
Epoch 1/5
145/145 [=====] - 12s 47ms/step - loss: 0.5855 - accuracy: 0.7915 - val_loss: 0.5020 - val_accuracy: 0.7951
Epoch 2/5
145/145 [=====] - 6s 38ms/step - loss: 0.3160 - accuracy: 0.8357 - val_loss: 0.4855 - val_accuracy: 0.8090
Epoch 3/5
145/145 [=====] - 5s 37ms/step - loss: 0.0160 - accuracy: 0.9990 - val_loss: 0.4881 - val_accuracy: 0.8090
Epoch 4/5
145/145 [=====] - 6s 38ms/step - loss: 0.0033 - accuracy: 0.9991 - val_loss: 0.4893 - val_accuracy: 0.8090
Epoch 5/5
145/145 [=====] - 6s 38ms/step - loss: 0.0024 - accuracy: 0.9992 - val_loss: 0.4898 - val_accuracy: 0.8090
73/73 [=====] - 0s 4ms/step - loss: 0.4898 - accuracy: 0.8090
Accuracy: 80.90%
```

*Figure 27: Résultats du modèle multicouches*

Les résultats du réseau de neurones multicouches montrent une diminution significative de l'erreur (Loss) au fur et à mesure des epochs. Cela indique que le modèle est capable d'apprendre efficacement à prédire la classe des tweets en ajustant les poids des neurones.

L'accuracy d'apprentissage montre également une augmentation progressive au fil des epochs, atteignant finalement une précision presque parfaite (supérieure à 0.99) lors des derniers epochs. Cela suggère que le modèle a pu bien généraliser et classer avec précision les tweets d'apprentissage.

L'accuracy de validation finale est de 80.90%, ce qui indique que le modèle est capable de prédire correctement la classe des tweets dans environ 81% des cas. Cependant, il est important de noter qu'il y a une légère différence entre l'accuracy d'apprentissage (presque parfaite) et l'accuracy de validation.

## 2) Description des expériences

Différents algorithmes de machine learning et de deep learning ont été testés pour la classification des tweets et ont montré des performances variables. Pour savoir quelle est la meilleure méthode nous avons jugé nécessaire de comparer les différents algorithmes entre eux.

### *Machine Learning*

Tout d'abord, nous avons fait une comparaison des algorithmes du machine learning à savoir les modèles *KNN*, *Arbre de décision*, *Régression logistique* et *SVM*. Nous avons noté que le **SVM** a une légère performance de plus que les autres algorithmes.

### *Deep Learning*

Ensuite, nous avons comparé les résultats du *réseau de neurones monocouche* et ceux du *réseau de neurones multicouches*. Après constat, on note que le **réseau de neurones multicouches** est plus performant.

### *Machine Learning vs Deep Learning*

Enfin, nous avons fait une comparaison du *SVM* et du *réseau de neurones multicouches*. On voit que le *SVM* et le *réseau de neurones multicouches* sont sensiblement égales en terme de performance.

Mais d'après nos recherches, si les données des tweets sont volumineuses et équilibrées, le meilleur choix parmi ces algorithmes serait le réseau de neurones multicouches.

Les réseaux de neurones multicouches sont connus pour leur capacité à capturer des relations complexes et non linéaires entre les variables d'entrée. Ils peuvent extraire des caractéristiques pertinentes à partir de données volumineuses et fournir une représentation riche des tweets. Ils montrent de bonnes performances en termes de généralisation sur de grandes quantités de données lorsqu'ils sont correctement entraînés. Si les données des tweets sont équilibrées, cela signifie qu'il y a un nombre égal d'exemples positifs et négatifs. Les réseaux de neurones multicouches n'ont pas de biais inhérent envers une classe particulière et peuvent apprendre de manière égale à partir des deux classes, ce qui est important pour une classification équilibrée et précise.

## IV. Conclusion

En somme, cette étude sur la fouille d'opinions nous a permis d'explorer différentes méthodes et techniques pour extraire, analyser et comprendre les opinions exprimées dans un ensemble de données textuelles. Les résultats obtenus ont démontré l'efficacité des algorithmes utilisés. La fouille d'opinions peut être utilisée de manière fiable et précise pour comprendre les opinions des populations et les utiliser dans la prise de décision, mais cela dépendra de plusieurs facteurs. Tout d'abord la qualité de données utilisée est cruciale, ensuite le choix et d'application d'algorithmes appropriés jouent un rôle clé dans l'exactitude des résultats. De plus, la validation et l'évaluation des résultats sont essentielles pour évaluer la fiabilité des conclusions obtenues. Enfin, il est crucial de prendre en compte le contexte et les nuances des opinions exprimées. Une analyse approfondie des résultats et une interprétation contextuelle sont nécessaires pour une utilisation précise des opinions dans la prise de décision.

Dans le domaine de la fouille d'opinions, plusieurs perspectives de recherche prometteuses telles que l'amélioration des techniques d'analyse sentimentale et l'analyse multilingue des opinions car les méthodes de fouille d'opinions sont principalement développées pour les langues anglophones. Il existe un besoin croissant de développer des techniques capables d'analyser et de comprendre les opinions dans d'autres langues.

## V. Références

- ✓ C.C. Aggarwal and C.X. Zhai (eds.), Mining Text Data
- ✓ Fouille d'opinions et analyse de sentiments | ATALA, Fouille d'opinions (cnrs.fr)
- ✓ Fouille d'opinions (cnrs.fr)
- ✓ actes\_deft2009.pdf (upsaclay.fr)
- ✓ Le cours et les TP du prof