



Universite Sultan Moulay Slimane
Ecole Nationale des Sciences Appliquées de Khouribga

REAL-TIME DATA PROCESSING FOR SENTIMENT ANALYSIS OF MOROCCAN DIALECT IN YOUTUBE

Réalisé par :

*El krem Abdelhamid
Elmehdi Ouassif*

Encadré Par :

Pr. Imad Hafidi

Table des matières

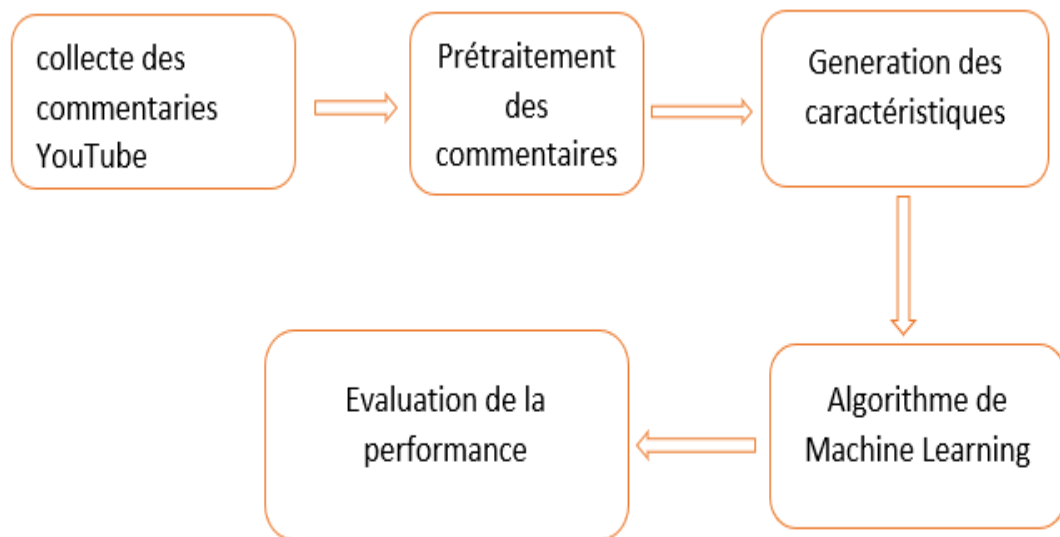
1	Introduction	2
2	Le processus d'analyse de sentiments	3
3	Les technologies et les méthodes utilisées	3
3.1	Les technologies utilisées :	3
3.2	Prétraitement de données en Python Cas normal :	10
3.3	Extraction de caractéristiques à partir de texte :	12
3.4	Algorithmes de classification :	14
4	Methodology d'analyse en temps réel :	14
4.1	Analyse des sentiments à l'aide de PySpark	14
4.2	traitement en temps reel avec le systeme de fichiers local . . .	19
4.2.1	1-Création du Streaming Context et réception des data streams	19
4.2.2	2-Exécution d'opérations sur les data streams	19
4.2.3	3-Démarrer le service de streaming	20
5	Conclusion	21

1 Introduction

L'analyse des sentiments est le processus de détection des sentiments positifs ou négatifs dans un texte. Il est souvent utilisé par les entreprises pour détecter les sentiments dans les données sociales, évaluer la réputation de la marque et comprendre les clients. L'analyse des sentiments se concentre sur la polarité (positive, négative, neutre) mais aussi sur les sentiments et les émotions (colère, heureux, triste, etc.), l'urgence (urgente, pas urgente) et même les intentions (intéressé contre pas intéressé). Selon la façon dont vous souhaitez interpréter les commentaires et les requêtes des clients, vous pouvez définir et adapter vos catégories pour répondre à vos besoins d'analyse des sentiments.

2 Le processus d'analyse de sentiments

Le processus d'analyse de sentiments des commentaires YouTube fonctionne en cinq étapes, comme le montre la figure ci-dessous.



3 Les technologies et les méthodes utilisées

3.1 Les technologies utilisées :

Apache Spark : définition

Apache Spark est une infrastructure de traitement de données pouvant effectuer rapidement des tâches de traitement sur des grands ensembles de données. Depuis ses modestes débuts à l'AMPLab de l'Université de Berkeley en 2009, il est devenu l'un des principaux frameworks de traitement distribué Big Data dans le monde. Ce système a également la faculté de répartir ces activités sur plusieurs ordinateurs, soit seul, soit en tandem avec d'autres outils informatiques distribués. Ces deux qualités sont essentielles dans le monde du big data et de l'apprentissage automatique. Ils nécessitent

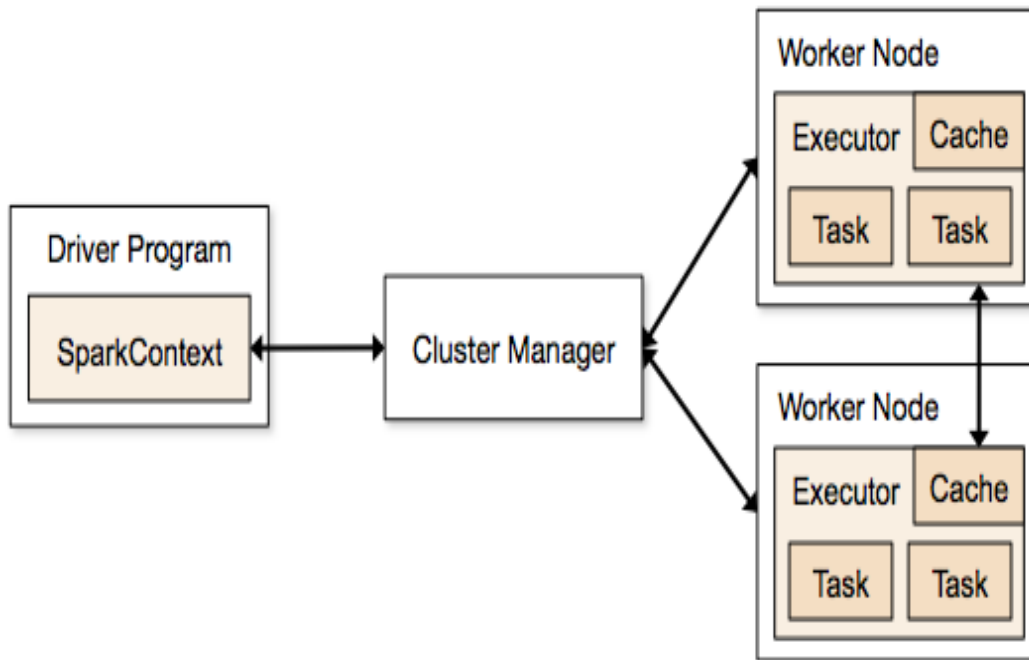
la mobilisation d'une puissance de calcul massive pour exploiter de grands ensembles de données.

Utilisation au quotidien :

Spark soulage les développeurs de certaines tâches de programmation grâce à une API facile à utiliser. Ce dernier permet d'éliminer une grande partie du travail fastidieux de l'informatique distribuée et du traitement des données volumineuses. Cette application est utilisée par les banques, les sociétés de télécommunications et des jeux, les gouvernements et surtout les géants de la technologie. Notamment Apple, Facebook, IBM et Microsoft.

Son architecture :

À un niveau fondamental, une application Apache Spark se compose : d'un pilote, qui convertit le code de l'utilisateur en plusieurs tâches pouvant être réparties sur les nœuds de travail, des exécuteurs, qui s'exécutent sur ces nœuds et effectuent les tâches qui leur sont assignées. Une certaine forme de gestionnaire de cluster est nécessaire pour assurer la médiation entre les deux. Il peut fonctionner en mode autonome de cluster qui nécessite simplement le framework Apache Spark et une JVM sur chaque machine. Cependant, il est plus probable que vous souhaitiez profiter d'un système de gestion des ressources ou de cluster plus robuste pour vous occuper de l'affectation des travailleurs à la demande. Dans une entreprise, cela signifie qu'il faut utiliser Hadoop YARN (c'est ainsi que les distributions Cloudera et Hortonworks exécutent les tâches Spark). Ce moteur d'analyses peut également fonctionner sur Mesos, Kubernetes et Docker Swarm.

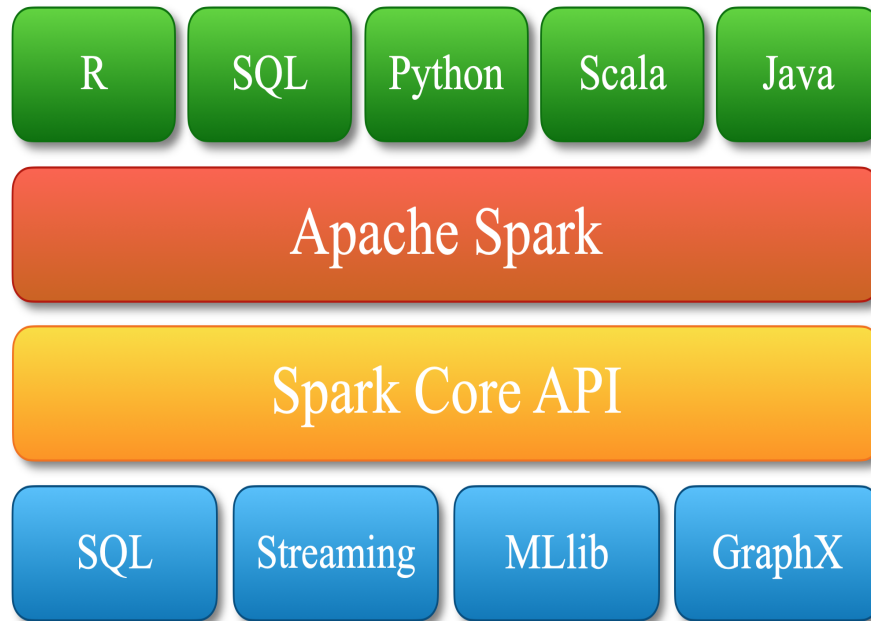


Autres fonctionnements du système :

Ce programme se trouve dans le cadre d'Amazon EMR, Google Cloud Dataproc et Microsoft Azure HDInsight pour les personnes qui recherchent une solution gérée. La société qui emploie les fondateurs de ce logiciel, propose également la Databricks Unified Analytics Platform. Cette dernière s'agit d'un service géré complet qui offre : des clusters Apache Spark, une prise en charge de la diffusion en continu, un développement de notebook intégré basé sur le Web, et des performances d'E / S cloud optimisées sur une distribution standard d'Apache Spark. Apache Spark intègre les commandes de traitement des données de l'utilisateur dans un graphique acyclique dirigé, ou DAG. Le DAG est la couche de planification déterminant quelles tâches sont exécutées sur quels nœuds et dans quelle séquence.

Composants de Spark :

Spark dans son ensemble se compose de diverses bibliothèques, API, bases de données, etc. Ses principaux composants sont les suivants :



1. Spark Core :

Spark Core est le bloc de base, qui comprend tous les composants pour la planification des tâches, l'exécution de diverses opérations de mémoire, la tolérance aux pannes, etc. Il héberge également l'API qui se compose de RDD. De plus, il fournit des API pour créer et manipuler des données dans RDD.

2. Spark SQL :

Apache Spark fonctionne avec les données non structurées à l'aide de son outil « aller à », Spark SQL. Ce dernier permet d'interroger des données via SQL, ainsi que via la forme SQL d'Apache Hive appelée Hive Query Language (HQL). Il prend également en charge les données de diverses sources telles que les tables d'analyse, les fichiers journaux, JSON, etc. Il permet aux programmeurs de combiner des requêtes SQL avec des modifications programmables ou des manipulations prises en charge par RDD en Python, Java, Scala et R.

3. Spark Streaming :

Cet outil traite les flux de données en direct. Des exemples de ces données incluent les fichiers journaux, les messages contenant des mises à jour de statut publiées par les utilisateurs, etc.

4. GraphX :

Spark GraphX est équipé d'une sélection d'algorithmes distribués pour le traitement des structures de graphes, y compris une implémentation du PageRank de Google.

Ces algorithmes utilisent l'approche RDD de Spark Core pour modéliser les données. Le package GraphFrames vous permet d'effectuer des opérations graphiques sur des dataframes, notamment en tirant parti de l'optimiseur Catalyst pour

5. MLlib :

Apache Spark propose une bibliothèque contenant des services communs de Machine Learning (ML) appelés MLlib. Il fournit différents types d'algorithmes de ML, notamment la régression, le clustering et la classification, qui peuvent effectuer diverses opérations sur les données pour en tirer des informations significatives.

PySpark :

PySpark est une interface pour Apache Spark en Python. Elle vous permet non seulement d'écrire des applications Spark à l'aide d'API Python, mais fournit également le shell PySpark pour analyser interactivement vos données dans un environnement distribué. PySpark supporte la plupart des fonctionnalités de Spark telles que Spark SQL, DataFrame, Streaming, MLlib (Machine Learning) et Spark Core. Pyspark est une alternative plus puissante que Pandas Python lorsqu'il s'agit du traitement de gros volume de données. Les principales caractéristiques qui font de Pyspark la librairie de traitement de données de Python la plus puissante sont les suivantes :

- **Calcul en temps réel** : PySpark permet le calcul en temps réel sur une grande quantité de données car il se concentre sur le traitement en mémoire. Il présente une faible latence.
- **Support de plusieurs langages** : la librairie PySpark est adapté à divers langages de programmation tels que Scala, Java, Python et R. Sa compatibilité en fait une librairie de choix pour le traitement d'énormes ensembles de données.
- **Mise en cache et constance du disque** : la librairie PySpark offre une mise en cache puissante et une bonne constance du disque.
- **Traitement rapide** : PySpark nous permet d'atteindre une grande vitesse de traitement des données, qui est environ 100 fois plus ra-

pide en mémoire et 10 fois plus rapide sur le disque.

- Les opérations dans un DataFrame PySpark sont paresseuses par nature (lazy evaluation) mais, dans le cas de pandas, nous obtenons le résultat dès que nous appliquons une opération.
- Dans un DataFrame PySpark, nous ne pouvons pas modifier le DataFrame en raison de sa propriété immuable, nous devons le transformer. Mais dans Pandas, ce n'est pas le cas.
- Les opérations complexes sont plus faciles à réaliser dans Pandas que dans un DataFrame PySpark.

Databricks :

Développée en 2013 par 7 anciens membres du projet Apache Spark, Databricks est une plateforme analytique Big Data proposant aux entreprises d'utiliser les technologies analytiques, l'intelligence artificielle et le Machine Learning pour réaliser des prédictions et prendre de meilleures décisions pour leur développement. La firme décrit sa solution comme une plateforme analytique unifiée, un espace de travail partagé permettant à différents types d'employés de collaborer sur des projets Big Data. Il s'agit d'une boîte à outils permettant aux ingénieurs de procéder au data cleaning ou au data access. Databricks regroupe aussi des outils analytiques dont Spark, et une fonctionnalité permettant de communiquer avec des data scientists ou des data analysts. Parmi les principaux clients de l'entreprise, on compte notamment Salesforce.com, Viacom, Shell, HP ou encore Hotels.com. Au total, la firme compte plus de 500 clients, certains payent plusieurs millions de dollars par an. Databricks fournit une plateforme unifiée et ouverte pour toutes vos données. Il offre aux data scientists, aux ingénieurs des données et aux analystes des données un environnement collaboratif simple pour exécuter des charges de travail d'analyse de données interactives et planifiées. Databricks fonctionne à 100 % sur une base Apache Spark. Par conséquent, tout code ou application développé sur Databricks peut s'exécuter sur n'importe quelle distribution compatible Apache Spark.

Koalas :

Certains défis subsistent encore. Par conséquent, lorsqu'il s'agit de traiter les big data et des cadres de traitement distribués, Spark devient le choix de facto pour la communauté des données. L'une des

nouvelles offres de Databricks est la bibliothèque open source bibliothèque open source appelée Koalas. Elle fournit une API Pandas DataFrame au-dessus d'Apache Spark. Par conséquent, elle tire parti de la mise en œuvre de DataFrames par Spark, de l'optimisation des requêtes et des connecteurs de sources de données, le tout dans un contexte où la qualité des données est primordiale. Il fournit une solution complète aux problèmes de qualité des données, y compris l'optimisation des requêtes et les connecteurs de sources de données, le tout avec la syntaxe Pandas. En substance, il offre une En substance, il offre une alternative intéressante pour utiliser la puissance de Spark tout en travaillant avec Pandas.

PyArabic :

Une bibliothèque spécifique à la langue arabe pour Python, qui fournit des fonctions de base pour manipuler les lettres et les textes arabes, comme la détection des lettres arabes, des groupes de lettres arabes et de leurs caractéristiques, la suppression des diacritiques, etc.

la tokenisation :

la tokenisation se réfère essentiellement à la division d'un plus grand corps de texte en lignes, mots plus petits ou même à la création de mots pour une langue non anglaise.

CountVectorizer :

CountVectorizer est un excellent outil fourni par la bibliothèque scikit-learn en Python. Il est utilisé pour transformer un texte donné en un vecteur sur la base de la fréquence (compte) de chaque mot qui apparaît dans l'ensemble du texte.

TF-IDF :

Le TF-IDF (de l'anglais term frequency-inverse document frequency) est une méthode de pondération souvent utilisée en recherche d'information et en particulier dans la fouille de textes. Cette mesure statistique permet d'évaluer l'importance d'un terme contenu dans un document, relativement à une collection ou un corpus. Le poids augmente proportionnellement au nombre d'occurrences du mot dans le document. Il varie également en fonction de la fréquence du mot dans le corpus.

3.2 Prétraitement de données en Python Cas normal :

1. La suppression des mots d'arrêt (StopWords)

```
stop_words=stop_word_darija+stop_word_en+stop_word_ar+stop_word_fr
def remove_stop_words(text):
    return " ".join([w for w in text.split() if w.lower() not in stop_words])
remove_stop_words('Tbarkellah alik nta dmaghhh thumbs_up red_heart')

mot="Sh7al bqety katfkar bash matnsash shy laqta mn laqtat li majitish m9erre3"
print(remove_stop_words(mot))

250
Sh7al bqety katfkar matnsash shy laqta laqtat majitish m9erre3
```

2. La Suppression des liens :

```
#remove URLS
import re
def remove_URLS(text):
    return re.sub(r'(http|https)\S+', '', text)

remove_URLS("https://www.youtube.com/watch?v=vgABU-gH-y8&feature=share khoti chaj3oni🙏")

' khoti chaj3oni🙏'
```

3. Suppression des caractères spéciaux et des signes de ponctuations :

```
#remove special caracteres

def remove_speciel_caracteres(text):
    return text.translate({ord(c): "" for c in ""!"#$%&'()*+,-./:;<=>@[\]^_`{|}~◆!:""})
remove_speciel_caracteres("Ana yallah bdit fel YouTube & bghitkom 3afakoum???" )

'Ana yallah bdit fel YouTube  bghitkom 3afakoum '
```

4. Suppression des commentaires contenant des publicités :

```
#remove comments containing publicity
import re
def remove_publicity_comments(text):
    p = '''#?(tabonaw|de3moni|abone|tde3moni|دعم|ابوني|بوлад|ابوني|تعارونوني|ابوني|ارلني|بدمعني|ارلني|abonnez vous)[a-z0-9]*'''
    return re.sub(p, '', text)

remove_publicity_comments("Ana yallah bdit fel YouTube ou bghitkom tde3moni 3afakoum")

'Ana yallah bdit fel YouTube ou bghitkom 3afakoum'
```

5. Normalisation : Dans cette étape, nous éliminons toutes les voyelles consécutives :

tbarklaah, JAMIIL هه funny

```
return comment

print(comment_text_preprocessor("حتى التعليق خصما يكون فيها نوع من الاحترام"))
```

```
#remove duplicate emojis

import emoji
from collections import Counter

def remove_duplicate_emojis(text):
    count=Counter("".join(c for c in text if c in emoji.UNICODE_EMOJI['en']))
    new_text=""
    for i in text:
        if(i in count):
            if(i not in new_text):
                new_text=new_text+" "+i
            else:
                new_text=new_text+i

    return new_text

remove_duplicate_emojis("Tbarkellaaah aliiik nta dmaaghhh👍👍👍👍👍👍👍👍👍👍❤️❤️❤️")
```

Activator Windows.

```
#replace emojis with their meaning|
#!pip install emoji --upgrade
import emoji

def replace_emojis(text):
    return emoji.demojize(text, delimiters=(" ", " "))
replace_emojis(remove_duplicate_emojis("Tbarkellaaah aliiik nta dmaaghhh 👍 ❤️"))

'Tbarkellaaah aliiik nta dmaaghhh   thumbs_up   red_heart '
```

3.3 Extraction de caractéristiques à partir de texte :

La première étape d'un classificateur de texte d'apprentissage automatique consiste à transformer l'extraction de texte ou la vectorisation de texte, et l'approche classique a été le sac de mots ou le sac de ngrammes avec leur fréquence.

Sac de mots :

Un modèle de sac de mots, ou BoW en abrégé, est un moyen d'extraire des caractéristiques du texte à utiliser dans la modélisation, par exemple avec des algorithmes d'apprentissage automatique. L'approche est très simple et flexible, et peut être utilisée de multiples façons pour extraire des caractéristiques de documents. Un sac de mots est une représentation de texte qui décrit l'occurrence de mots dans un document. Cela implique deux choses :

- (a) Un vocabulaire de mots connus.
- (b) Une mesure de la présence de mots connus.
- Étape 1 : collecter des données.
- Étape 2 : Concevoir le vocabulaire : Dans cette étape on fait une liste de tous les mots de notre vocabulaire modèle.
- Étape 3 : Créer des vecteurs de document : L'étape suivante consiste à marquer les mots dans chaque document.

L'objectif est de transformer chaque document de texte libre en un vecteur que nous pouvons utiliser comme entrée ou sortie pour un modèle d'apprentissage automatique.

```
#Bag of words
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(min_df=0.0, max_df=1.0, ngram_range=(1, 1))
X_traincv = cv.fit_transform(X_train['Clean comment']).toarray()
X_traincv = pd.DataFrame(X_traincv, columns=cv.get_feature_names())

X_testcv = cv.transform(X_test['Clean comment']).toarray()
X_testcv = pd.DataFrame(X_testcv, columns=cv.get_feature_names())
X_traincv.head()
```

	059	100100	11	15	1k	39l	39lhom	3adime	3aila	3ajaza	...	يكر	يكي	يكونوا	يلا	يمك	ينشر	ينه	يوتوب	يوتيوب	يوم
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

3.4 Algorithmes de classification :

Régression logistique :

	precision	recall	f1-score	support
-1.0	0.93	1.00	0.97	83
1.0	1.00	0.82	0.90	34
accuracy			0.95	117
macro avg	0.97	0.91	0.93	117
weighted avg	0.95	0.95	0.95	117

	precision	recall	f1-score	support
-1.0	1.00	1.00	1.00	287
1.0	1.00	1.00	1.00	62
accuracy			1.00	349
macro avg	1.00	1.00	1.00	349
weighted avg	1.00	1.00	1.00	349

	0	1
0	83	0
1	6	28

4 Methodology d'analyse en temps réel :

Nous avons fait une analyse avec un calcul distribuer en utilisant pyspark et ensuite, nous avons fait un exemple de traitement en temps réel avec le système de fichiers local.

4.1 Analyse des sentiments à l'aide de PySpark

Le jeu de données que nous allons utiliser est un échantillon des données de commentaires de youtube .Nous allons prédire le sentiment de l'avis donné (positif ou négatif). Commençons par lire les données textuelles et créer un Dataframe Spark :

```
# File location and type
```

- ▶ (3) Spark Jobs

Showing all 518 rows.

```
root
|-- sentence: string (nullable = true)
|-- polarity: integer (nullable = true)
```

15

Nous avons expliqué l'utilisation de koalas dans la section "technologies utilisées" 0.3.1

L'étape suivante consiste à commencer le pre-traitement

```
import pyarabic.araby as araby
import re
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words = set(stopwords.words('arabic'))
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

def pre_pros(sentence):
    sentence = sentence.lower()
    #Supprimer les liens url
    sentence =
        re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|(?:%[0-9a-fA-F][0-9a-fA-F]))',
            '', sentence)
    #Supprimer les mots non ecrits garder que les mots
    sentence = re.sub("[^a-zA-Z0-9- ]", "", sentence)
    # supprimer les espaces > 1
    sentence = re.sub(' +', ' ', sentence)
    #Supprimer plusieurs lettres repetant des mots
    sentence = re.sub(r'(\.)\1+', r'\1\1', sentence)
    #Supprimer les stop_words
    sentence1 = [word for word in sentence.split() if
        word.lower() not in stop_words]
    sentence = " ".join(sentence1)
    sentence = ' '.join(word for word in sentence.split() if
        word not in stop_words)
    #garder que les mots qui n est pas en anglais ou en
    francais
    if (sentence != "" and detect(sentence)!="en" and
        detect(sentence)!="fr"):
        #Suppression de "At-tachkil"
        sentence= araby.strip_diacritics(sentence)
    sentence = "".join(sentence)
    return sentence.strip()
```


	pre_sentence	polarity	sentence
0	فيديو رائع انا أشد المعجبين	1	...واو فيديو رائع انا إلهام و انا من أشد المعجبين
1	sh7al bqety katfkar bash matnsash shy laqta mn...	1	Sh7al bqety katfkar bash matnsash shy laqta mn...
2	طلعتي كمشيه زياش	1	👍👍 طلعتي كمشيه زياش
3	yallah bdit fel youtube ou bghitkom tde3moni a...	1	Ana yallah bdit fel YouTube ou bghitkom tde3mo...
4	tbarkellaah aliiik dmaaghh	1	Tbarkellaah aliiik dmaaghhh👍👍👍👍👍👍👍👍❤️❤️...

L'étape suivante consiste à lancer le processus de tokenisation.

Puisque nous traitons maintenant uniquement des tokens au lieu d’une

```
len_udf = udf(lambda s: len(s), IntegerType())
refined_text_df =
    tf_idf_df.withColumn("token_count", len_udf(col('tokens')))
refined_text_df.orderBy(rand()).show(10)
```

17

```
count_vec=CountVectorizer(inputCol='tokens',outputCol='features')
cv_text_df=count_vec.fit(refined_text_df).transform(refined_text_df)
cv_text_df.select(['tokens','token_count','features','polarity']).show(10)
```

Nous pouvons utiliser n'importe quel modèle de classification sur ces données, mais nous commençons par former un modèle de régression logistique :

```
from pyspark.ml.classification import LogisticRegression
training_df,test_df=model_text_df.randomSplit([0.75,0.25])
```

Pour vérifier la présence de suffisamment d'enregistrements pour les deux classes dans les ensembles de formation et de test, nous pouvons appliquer la fonction groupBy sur la colonne Label :

```
training_df.groupBy('polarity').count().show()
test_df.groupBy('polarity').count().show()
```

► (3) Spark Jobs

```
+-----+-----+
|polarity|count|
+-----+-----+
|         1|   14|
|         0|   59|
+-----+-----+
```

► (3) Spark Jobs

```
+-----+-----+
|polarity|count|
+-----+-----+
|         1|   58|
|         0|  245|
+-----+-----+
```

```
log_reg=LogisticRegression(featuresCol='features_vec',labelCol='polarity')
    .fit(training_df)
results=log_reg.evaluate(test_df).predictions
results.show(30)
```

```
training_predictions=log_reg.evaluate(training_df)
print(training_predictions.accuracy)
test_results=log_reg.evaluate(test_df)
print(test_results.accuracy)
```

4.2 traitement en temps reel avec le systeme de fichiers local

4.2.1 1-Création du Streaming Context et réception des data streams

Streaming Context : est le principal point d'entrée de toute application de streaming. Il peut être créé en instanciant la classe `StreamingContext` du module `pyspark.streaming`.

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
```

En créant le `StreamingContext`, nous pouvons spécifier la durée du lot, par exemple ici la durée du lot est de 3 secondes.

```
sc = SparkContext(appName = "Text Cleaning")
strc = StreamingContext(sc, 3)
```

Une fois le `StreamingContext` créé, nous pouvons commencer à recevoir des données sous forme de `DStream` via le protocole TCP sur un port spécifique. Par exemple, ici le nom d'hôte est spécifié comme "localhost" et le port utilisé est 8084.

```
text_data = strc.socketTextStream("localhost", 8084)
```

4.2.2 2-Exécution d'opérations sur les data streams

Après avoir créé un objet `DStream`, nous pouvons effectuer des opérations sur celui-ci en fonction de nos besoins. Ici, nous avons écrit une fonction personnalisée de nettoyage de texte.

Cette fonction convertit d'abord le texte d'entrée en minuscules, puis supprime les espaces supplémentaires, les caractères non alphanumériques,

les liens/URL, les mots d'arrêt, et enfin lemmatise le texte en utilisant la bibliothèque NLTK.

```
def pre_pros(sentence):
    sentence = sentence.lower()
    #Supprimer les liens url
    sentence =
        re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|(?:%[0-9a-fA-F][0-9a-fA-F]))',
            '', sentence)
    #Supprimer les mots non crits - ne garder que les mots
    sentence = re.sub("[^a-zA-Z0-9- ]", " ", sentence)

    # supprimer les espaces > 1
    sentence = re.sub(' +', ' ', sentence)
    #Supprimer plusieurs lettres rptant des mots
    sentence = re.sub(r'(\.|\1+)', r'\1', sentence)

    #garder que les mots qui n'est pas en anglais ou en
    francais
    if (sentence != "" and detect(sentence)!="en" and
        detect(sentence)!="fr"):
        #Suppression de "At-tachkil"
        sentence= araby.strip_diacritics(sentence)
    sentence = " ".join(sentence)
    return sentence.strip()
```

4.2.3 3-Démarrer le service de streaming

Le service de streaming n'a pas encore démarré. Utilisez la fonction `start()` au dessus de l'objet `StreamingContext` pour le démarrer et continuer à recevoir des données en streaming jusqu'à ce que la commande de terminaison (`Ctrl + C` ou `Ctrl + Z`) ne soit pas reçue par la fonction `awaitTermination()`.

```
strc.start()
strc.awaitTermination()
```

Nous devons d'abord exécuter la commande `'nc'` (utilitaire Netcat) pour envoyer les données textuelles du serveur de données au serveur

de streaming spark. Netcat est un petit utilitaire disponible sur les systèmes de type Unix pour lire et écrire sur des connexions réseau utilisant des ports TCP ou UDP. Ses deux options principales sont -

- l : Pour permettre à nc d'écouter une connexion entrante plutôt que d'initier une connexion à un hôte distant.
- k : Force nc à rester à l'écoute d'une autre connexion après que sa connexion actuelle soit terminée.

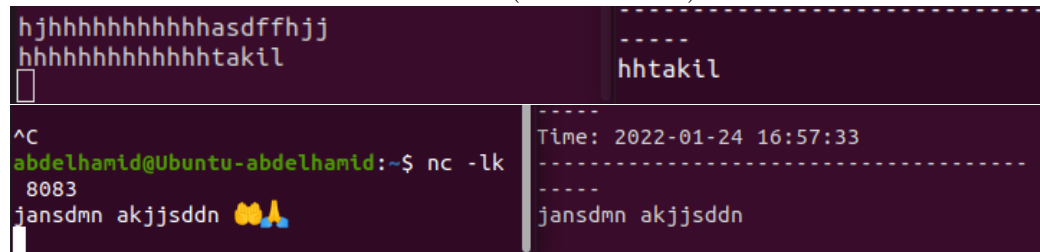
Exécutez donc la commande nc suivante dans le terminal.

```
nc -lk 8083
```

De même, exécutez le script pyspark dans un autre terminal en utilisant la commande suivante afin d'effectuer un nettoyage de texte sur les données reçues.

```
spark-submit streaming.py localhost 8083
```

Dans cette démo, tout texte écrit dans le terminal (exécutant le serveur netcat) sera nettoyé et le texte nettoyé sera imprimé dans un autre terminal toutes les 3 secondes (durée du lot).



The image shows two terminal windows side-by-side. The left window is a netcat listener running 'nc -lk 8083'. It receives a connection and shows incoming text: 'hjhjhjhjhjhjhhasdffhjj' and 'hhjhjhjhjhjhhtakil'. The right window shows the output of the 'streaming.py' script, which receives the text from the netcat listener and displays it after cleaning: 'hhtakil'. The script also shows a timestamp 'Time: 2022-01-24 16:57:33' and the cleaned text 'jansdmn akjjsddn' with some emojis.

5 Conclusion

Ce projet était très enrichissant, car il nous a permis de se familiariser avec le domaine de l'analyse de sentiment pour le dialecte marocain, nous avons rencontré la difficulté de l'analyse de sentiment pour le dialecte marocain qui est une langue non structurée et complique davantage l'analyse des sentiments, car la majorité des outils de NLP existants ont été développés sur la base de la norme arabe moderne, ce projet nous a permis aussi de traiter et analyser les données en temps réel en utilisant les technologies Big Data.

Références

- [1] :Machine Learning with PySpark
<https://doi.org/10.1007/978-1-4842-7777-5>
- [2] :A hybrid approach to translate Moroccan Arabic dialect
- [3] :Social media Sentiment Monitoring in Smart Cities An application to Moroccan dialects
- [4] :Machine Learning Techniques in Storm
- [5] :Sentiment analysis for moroccan dialect
- [6] :Twitter Sentiment Analysis On Coronavirus Outbreak Using Machine Learning Algorithms
- [7] :Development of Real Time Analytics of Movies Review Data using PySpark
- [8] :Mining Tweets of Moroccan Users using the Framework Hadoop, NLP, K-means and Basemap
- [9] :An Experimental Study on Sentiment Classification of Algerian Dialect Texts
- [10] :Sentiment Analysis using Neural Networks : A New Approach
- [11] :A Near Real-Time Approach for Sentiment Analysis Approach Using Arabic Tweets
- [12] :Emotion and sentiment analysis from Twitter text
- [13] :Big Data Real-time Processing Based on Storm
- [14] :A Framework for Sentiment Analysis with Opinion Mining of Hotel Reviews
- [15] :Spark Real Time Streaming — Real time Data Streaming With Apache Spark
<https://www.analyticsvidhya.com/blog/2021/06/real-time-data-streaming-using-apache-spark/> Accessed : 2022-01-25
- [16] :Apache Spark Structured Streaming with Pyspark — by Sercan Karagoz — Analytics Vidhya — Medium
<https://medium.com/analytics-vidhya/apache-spark-structured-streaming-with-pyspark-b4a054a7947d>

- [17] :5 Steps to Converting Python Jobs to PySpark —
by Mohini Kalamkar — HashmapInc — Medium
<https://medium.com/hashmapinc/5-steps-to-converting-python-jobs-to-pyspark-4b9988ad027a>
- [18] : - <https://arabicprogrammer.com/article/87412068214/>
- [19] :PySpark Drop Rows with NULL or None Values — Spark-ByExamples <https://sparkbyexamples.com/pyspark/pyspark-drop-rows-with-null-values/>
- [20] :fabriceyh/emoji_translate : *Translateyouremojis*[https://github.com/fabriceyh/emoji_translate](https://github.com/fabriceyh/emoji_ttranslate)