



RECURRENT NEURAL NETWORKS

BIG DATA MINIG

NOURINE Billel

TOUATI Islem

OUCHENE Souhil

Département : Génie Industriel

Spécialité : Data Science et Intelligence Artificielle

Année universitaire : 2022/2023

RESUME

Le but de ce projet est d'expliquer les fondements du Deep Learning, en particulier l'algorithme RNN, et développer une application de classification de sentiments sur le jeu de données Sentiment140 en utilisant PySpark

SOMMAIRE :

|. Introduction

||. Deep Learning

- 1.Définition
- 2.Domaine d'application

|||. Pyspark

IV. Les caractéristiques du dataset Sentiment140 datasetwith 1.6 million tweets

V. Recurrent Neural NETWORKS. (RNN):

- 1.Définition
- 2.Exemple RNN (traduction)
- 3.Les types du RNN

VI. Approche de résolution du problème de classification du jeu de données Sentiment140 en utilisant un RNN avec Pyspark

- 1.Création de Session pyspark
- 2.Chargement et visualisation des données
- 3.Nettoyage des données
- 4.Sentiment Analysis
 - 4.1. Tokenazation
 - 4.2. Remove Stopwords
 - 4.3. Lemmatization
 - 4.4. Vectorization

VII. Création et évaluation du modèle

- 1.Deviser le jeu de données en 2 ensembles Train et Test
- 2.Crétion du modèle RNN
3. Compilation du modèle
4. Evaluation du modèle sur les données du test
- 5.Les résultats obtenus

VIII. Problèmes rencontres

IX. Observation de l'environnement Spark

X. Conclusion

I- Introduction :

L'intelligence artificielle est une discipline scientifique recherchant des méthodes de solution de problèmes à forte complexité logique ou algorithmique. L'apprentissage automatique est un champ d'étude de l'intelligence artificielle. Par conséquent, L'apprentissage profond (en anglais Deep Learning, Deep Structured Learning, Hierarchical Learning) est un ensemble de méthodes d'apprentissage automatique tentant de modéliser avec un haut niveau d'abstraction des données grâce à des architectures articulées de différentes transformations non linéaires.

II - Deep Learning :

1. Définition

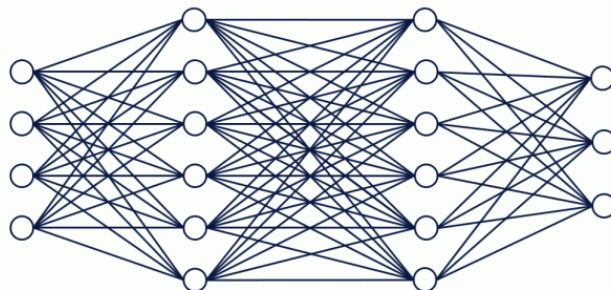
L'apprentissage profond (« Deep Learning ») est un ensemble de techniques d'apprentissage automatique qui a permis des avancées importantes en intelligence artificielle dans les dernières années. L'utilité essentielle du Deep Learning est le traitement en masse de données pour augmenter la performance d'un algorithme et rendre un système de plus en plus intelligent et autonome

Dans l'apprentissage automatique, un programme analyse un ensemble de données afin de tirer des règles qui permettront de tirer des conclusions sur de nouvelles données.

L'apprentissage profond est basé sur ce qui a été appelé, par analogie, des « réseaux de neurones artificiels », composés de milliers d'unités (les « neurones ») qui effectuent chacune de petites opérations simples. Les résultats d'une première couche de « neurones » servent d'entrée aux calculs d'une deuxième couche et ainsi de suite.

Par exemple, pour la reconnaissance visuelle, des premières couches d'unités identifient des lignes, des courbes, des angles... des couches supérieures identifient des formes, des combinaisons de formes, des objets, des contextes...

Les progrès de l'apprentissage profond ont été possibles notamment grâce à l'augmentation de la puissance des ordinateurs et au développement de grandes bases de données (« big data »)



2. Domaines d'application :

Ces techniques se développent dans le domaine de l'informatique appliquée aux NTIC (reconnaissance visuelle — par exemple d'un panneau de signalisation par un robot ou une voiture autonome — et vocale notamment) à la robotique, à la bio-informatique, la reconnaissance ou

comparaison de formes, la sécurité, la santé, etc..., la pédagogie assistée par l'informatique, et plus généralement à l'intelligence artificielle. L'apprentissage profond peut par exemple permettre à un ordinateur de mieux reconnaître des objets hautement déformables et/ou analyser par exemple les émotions révélées par un visage photographié ou filmé, ou analyser les mouvements et position des doigts d'une main, ce qui peut être utile pour traduire le langage des signes, améliorer le positionnement automatique d'une caméra, etc... Elles sont utilisées pour certaines formes d'aide au diagnostic médical (ex. : reconnaissance automatique d'un cancer en imagerie médicale), ou de prospective ou de prédiction (ex. : prédiction des propriétés d'un sol filmé par un robot).

III- Pyspark

Pyspark est une bibliothèque open source de traitement distribué de données en mémoire qui permet de traiter des données massives à grande échelle. Elle est développée par Apache Software Foundation et est basée sur le Framework Apache Spark.

Pyspark utilise le langage de programmation Python comme interface pour accéder aux fonctionnalités du Framework Spark. Il permet ainsi d'écrire des programmes distribués pour le traitement de données massives en utilisant la syntaxe et les fonctionnalités de Python. Pyspark fournit également des outils pour la manipulation de données, la gestion de cluster, la visualisation de données, l'apprentissage automatique, etc.

Pyspark peut être utilisé pour diverses tâches telles que l'analyse de données, la transformation de données, la modélisation prédictive, la classification de données, l'apprentissage en profondeur, etc. Il est très populaire dans le domaine du Big Data et est utilisé dans de nombreuses entreprises pour le traitement de données massives.



IV - les caractéristiques du dataset Sentiment140 dataset with 1.6 million tweets :

Le dataset Sentiment140 est un ensemble de données de tweets qui a été collecté en février 2009 à partir de Twitter avant qu'il ne limite l'accès aux données de son API. Il a été créé par Stanford University et est souvent utilisé dans les tâches de classification de sentiments. Il contient environ 1,6 million de tweets en anglais, qui ont été annotés avec leur polarité de sentiment (positif ou négatif).

Chaque tweet du dataset Sentiment140 est étiqueté avec un score de polarité de sentiment qui peut être 0 (négatif) ou 4 (positif), et le texte du tweet est également inclus. Le dataset a été divisé en un ensemble d'entraînement de 1,6 million de tweets et un ensemble de test de 498 tweets. Ce dataset a été largement utilisé dans la recherche en classification de sentiments et dans le développement de modèles d'apprentissage automatique pour la classification de sentiments en utilisant des techniques telles que le bag-of-words, les modèles de langage et l'apprentissage profond. Les résultats ont montré que les performances des modèles de classification de sentiments entraînés sur le dataset Sentiment140 sont comparables ou supérieures à celles des modèles entraînés sur d'autres datasets de classification de sentiments.

Le dataset contient les 6 champs suivants :

Target : la polarité du tweet (0 = négatif, 4 = positif)

ids : l'identifiant du tweet (2087)

date : la date du tweet (sam. mai 16 23:58:44 UTC 2009)

flag : la requête (lyx). Si aucune requête n'est spécifiée, cette valeur est NO_QUERY.

user : l'utilisateur qui a tweeté (robotickilldozr)

text : le contenu du tweet (Lyx est cool)



Negative



Neutral



Positive

V - Recurrent Neural NETWORKS. (RNN):

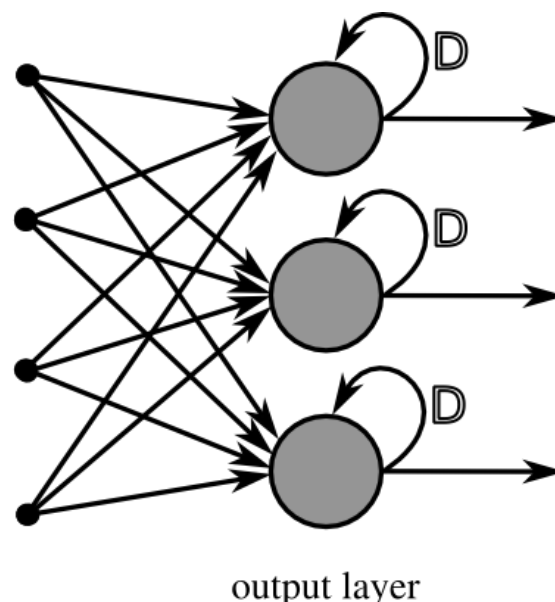
1.Définition:

Les réseaux de neurones récurrents (RNN) sont un type de réseau de neurones artificiels utilisé pour traiter des données séquentielles telles que des séquences de mots, de sons ou de données temporelles. Contrairement aux réseaux de neurones classiques qui traitent les données d'entrée couche par couche, les RNN ont des connexions récurrentes qui leur permettent de conserver une mémoire interne ou un état caché qui est mis à jour à chaque étape de la séquence. Cet état caché

permet aux RNN de traiter des données séquentielles en prenant en compte le contexte des données précédentes pour la prédiction des données futures.

Les RNN sont souvent utilisés pour des tâches telles que la reconnaissance de la parole, la génération de texte, la traduction automatique, la prédiction de la série temporelle, et d'autres tâches de traitement de séquences de données. Les RNN sont particulièrement utiles pour traiter des données de séquences de longueur variable, car ils peuvent traiter des séquences de données de longueur arbitraire.

La structure d'un RNN introduit un mécanisme de mémoire des entrées précédentes qui persiste dans les états internes du réseau et peut ainsi impacter toutes ses sorties futures. Avec cette simple modification il est en théorie possible d'approximer n'importe quelle fonction qui transforme une séquence d'entrée en une séquence de sortie donnée avec une précision arbitraire. Ce que ne permet pas le MLP. Le revers de la médaille est que ce type de RNN peut être particulièrement difficile à entraîner



2. RNN exemple (Traduction) :

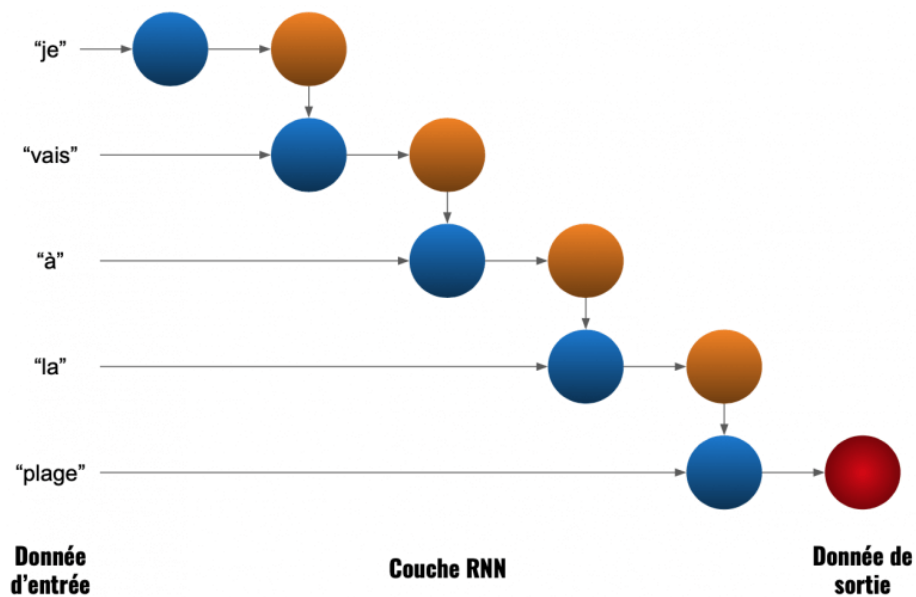
Les couches RNN sont récurrentes, c'est à dire que les informations qu'elles calculent vont être stockés en mémoire pour être réutiliser sur un prochain calcul.

Dans une couche classique non récurrente un neurone effectue un produit vectoriel entre l'entrée qu'il reçoit et son poids puis ajoute un biais. Ce calcul va ensuite passer dans sa fonction d'activation.

Dans une couche RNN un deuxième produit vectoriel contenant les données en mémoire vient s'ajouter au calcul. Ainsi, lorsqu'on veut prédire la valeur d'une action à un temps $t+1$, le modèle RNN prend en compte les précédentes valeurs dans sa prédiction.

En quelques sorte, le modèle comprend la variation qui a eu lieu entre $t-2$ et $t-1$, entre $t-1$ et t et peut prédire la future variation entre t et $t+1$

Un schéma sur un exemple de NLP sera peut-être plus précis. Ici, on calcul la signification de la phrase « je vais à la plage »



3. Les types du RNN:

One-to-One :

Le type le plus simple de réseau de neurones récurrent (RNN) est le "One-to-One", qui permet une entrée et une sortie unique. Il a des tailles d'entrée et de sortie fixes et agit comme un réseau neuronal traditionnel. L'application "One-to-One" peut être trouvée dans la classification d'images.

One-to-Many :

"One-to-Many" est un type de RNN qui produit plusieurs sorties lorsqu'il reçoit une seule entrée. Il prend une entrée de taille fixe et produit une séquence de sorties de données. Ses applications peuvent être trouvées dans la génération de musique et la description d'images.

Many-to-One :

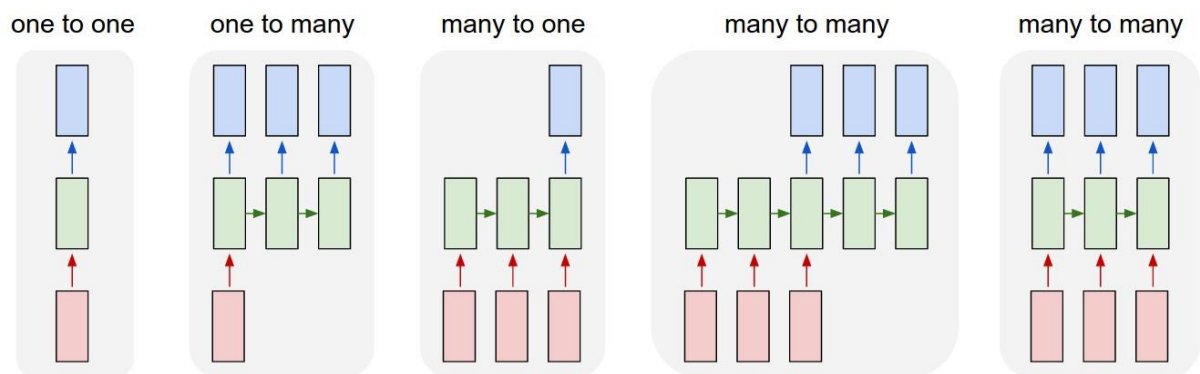
"Many-to-One" est utilisé lorsqu'une seule sortie est requise à partir de plusieurs unités d'entrée ou d'une séquence de celles-ci. Il prend une séquence d'entrées pour afficher une sortie fixe. L'analyse de sentiment est un exemple courant de ce type de réseau de neurones récurrents.

Many-to-Many :

"Many-to-Many" est utilisé pour générer une séquence de données de sortie à partir d'une séquence d'unités d'entrée. Ce type de RNN est ensuite divisé en deux sous-catégories :

Taille d'unité égale : Dans ce cas, le nombre d'unités d'entrée et de sortie est le même. Une application courante peut être trouvée dans la reconnaissance d'entités nommées (name-entity recognition).

Taille d'unité inégale : Dans ce cas, les entrées et les sorties ont des nombres d'unités différents. Son application peut être trouvée dans la traduction automatique (machine translation).



VI. Approche de résolution du problème de classification du jeu de données Sentiment140 en utilisant un RNN avec Pyspark :

1.Création de Session pyspark :

Créer une session PySpark est la première étape pour interagir avec les données dans un environnement PySpark. La session est essentiellement une porte d'entrée pour accéder aux fonctionnalités de PySpark telles que la lecture, l'écriture et la manipulation de données.

Lorsqu'une session PySpark est créée, elle initialise un environnement Spark, configure les paramètres de configuration de Spark, crée un contexte Spark et alloue les ressources nécessaires pour exécuter les tâches. Cela permet aux utilisateurs d'interagir avec les données de manière distribuée et de profiter de la puissance de traitement parallèle de Spark.

2.Chargement et visualisation des données :

Après avoir récupéré le jeu de données à partir d'un fichier CSV, la première étape de notre projet était d'observer les différentes caractéristiques de notre dataset :

- Les dimensions du dataset (nombre de lignes et de colonnes)
- Des informations sur les attributs du dataset (types, valeurs uniques...)
- Les valeurs manquantes dans le dataset

3.Nettoyage des données :

Le nettoyage des données est une étape importante dans l'analyse de données car il permet de supprimer les données inutiles, corrompues ou redondantes, ce qui peut améliorer la qualité des données et augmenter la précision des résultats obtenus lors de l'analyse des données. Il peut également aider à éviter les erreurs et les biais qui peuvent être introduits lors de l'analyse de données brutes et incomplètes.

Les étapes suivies pour nettoyer nos données sont les suivantes :

- Enlever les balises HTML

- Enlever les mentions
- Enlever les chiffres du texte
- Enlever les emojis
- Enlever les lettres isolées

4.Sentiment Analysis :

La Sentiment analysis, est une technique de traitement automatique du langage naturel qui consiste à déterminer le ton émotionnel d'un texte. Elle vise à identifier et extraire les opinions, les émotions et les sentiments exprimés dans un texte donné, tels que la positivité, la négativité ou la neutralité. Cette technique est utilisée dans de nombreux domaines, tels que le marketing, la politique, la finance et la recherche, pour comprendre les attitudes et les opinions des consommateurs, des électeurs, des investisseurs, etc

Les étapes suivies lors de notre projet pour effectuer l'analyse de sentiments sont :

4.1. Tokenazation :

La tokenization est une étape de prétraitement des données qui consiste à diviser un texte en unités discrètes appelées "tokens". Ces tokens peuvent être des mots, des phrases, des paragraphes ou des symboles de ponctuation.

4.2. Remove Stopwords :

C'est une étape de prétraitement des données dans le traitement automatique du langage naturel. Les stopwords sont des mots très courants qui n'apportent généralement pas beaucoup de sens ou d'informations utiles pour l'analyse des données textuelles L'étape de "remove stopwords" consiste à supprimer ces mots des données textuelles afin de se concentrer sur les mots plus importants et significatifs pour l'analyse. Cette étape permet de réduire la complexité des données textuelles et d'améliorer la précision de l'analyse.

4.3. Lemmatization

La lemmatisation est une technique de traitement de texte qui consiste à réduire les mots d'un texte à leur forme canonique, appelée "lemme". Cela implique de prendre en compte la morphologie des mots et de les ramener à leur forme de base, afin de faciliter leur analyse.

4.4. Vectorization

La vectorisation est le processus de conversion de données textuelles en vecteurs de nombres. Dans le cadre de l'analyse de texte, chaque document est représenté par un vecteur de nombres qui capture les caractéristiques importantes du document. La vectorisation permet aux algorithmes de traitement automatique du langage naturel (NLP) de traiter des données textuelles en utilisant des méthodes de l'apprentissage automatique.

VII. Création et évaluation du modèle :

La création du modèle du Deep Learning (RNN) passe par plusieurs étapes :

1. Déviser le jeu de données en 2 ensembles Train et Test :

Lorsqu'on travaille sur un projet d'apprentissage automatique, il est important de disposer de données à la fois pour entraîner notre modèle et pour évaluer sa performance. Ainsi, on divise notre ensemble de données en deux parties : l'ensemble d'entraînement (80%) et l'ensemble de test (20%).

2. Création du modèle RNN :

L'architecture de ce notre réseau de neurones récurrents (RNN) est assez simple. Voici une explication plus détaillée de chaque couche :

Couche RNN : Cette couche est une couche récurrente simple (SimpleRNN) qui contient 32 unités (ou neurones). L'activation utilisée est ReLU (Rectified Linear Unit), qui est une fonction d'activation couramment utilisée dans les réseaux de neurones profonds. L'entrée de cette couche est de forme (1,50), ce qui signifie qu'elle attend une séquence d'entrée de 50 mots (ou tokens) pour chaque observation dans le jeu de données.

Couche de sortie binaire : Cette couche est une couche dense qui contient une seule unité avec une fonction d'activation sigmoïde. La fonction sigmoïde est couramment utilisée pour les problèmes de classification binaire car elle renvoie une valeur comprise entre 0 et 1, qui peut être interprétée comme la probabilité que l'observation appartienne à la classe positive.

3. Compilation du modèle :

Une fois que le modèle est créé, il doit être compilé. Les paramètres utilisés pour notre compilation :

- **optimizer='adam'** : spécifie l'algorithme d'optimisation à utiliser pour ajuster les poids du modèle lors de l'apprentissage. Ici, l'optimiseur utilisé est Adam, qui est un algorithme d'optimisation stochastique couramment utilisé pour les réseaux de neurones.
- **loss='binary_crossentropy'** : spécifie la fonction de perte à utiliser pour mesurer l'erreur entre les prédictions du modèle et les valeurs de sortie réelles. Binary_crossentropy est une fonction de perte couramment utilisée pour les problèmes de classification binaire.
- **metrics=['accuracy']** : spécifie les métriques à utiliser pour évaluer les performances du modèle pendant l'apprentissage. Ici, la métrique utilisée est la précision (accuracy), qui mesure la proportion de prédictions correctes parmi toutes les prédictions effectuées.

4. Evaluation du modèle sur les données du test :

L'évaluation du modèle est une étape importante dans le processus d'apprentissage automatique, car elle permet de mesurer les performances du modèle et de déterminer s'il est suffisamment précis pour répondre aux exigences du problème

5. Les résultats obtenus :

Après avoir entraîné le modèle et évalué sa performance sur les données de test, nous avons obtenu les résultats suivants :

```
[ ] # Entraînement du modèle
model.fit(x_train, y_train, epochs=5, batch_size=64) # You can adjust the number of epochs and batch size to your needs
```

```
Epoch 1/5
2000/2000 [=====] - 7s 3ms/step - loss: 0.5971 - accuracy: 0.6741
Epoch 2/5
2000/2000 [=====] - 6s 3ms/step - loss: 0.5759 - accuracy: 0.6919
Epoch 3/5
2000/2000 [=====] - 5s 2ms/step - loss: 0.5697 - accuracy: 0.6984
Epoch 4/5
2000/2000 [=====] - 6s 3ms/step - loss: 0.5660 - accuracy: 0.7014
Epoch 5/5
2000/2000 [=====] - 5s 3ms/step - loss: 0.5634 - accuracy: 0.7039
<keras.callbacks.History at 0x7f16a04e0f10>
```

```
[ ] # Evaluation du modèle sur les données test
loss, accuracy = model.evaluate(x_test, y_test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

```
1000/1000 [=====] - 2s 2ms/step - loss: 0.5625 - accuracy: 0.7024
Test Loss: 0.5625287890434265
Test Accuracy: 0.7024062275886536
```

Train Accuracy : 0.7039

Test Accuracy : 0.7024

VIII. Problèmes rencontrés :

Lors de la réalisation de ce projet, nous avons rencontré plusieurs problèmes, notamment lors de l'utilisation de la méthode 'Word2Vec' dans PySpark, qui permet de représenter les mots sous forme de vecteurs numériques. L'implémentation distribuée de l'algorithme Word2Vec dans PySpark s'est avérée être un défi en raison de son temps d'exécution élevé. De plus, l'absence de packages pour le Deep Learning dans PySpark nous a contraints à convertir le DataFrame PySpark en un tableau NumPy pour ensuite utiliser TensorFlow. Cependant, cette conversion s'est révélée très exigeante en termes de puissance de calcul, ce qui nous a amenés à réduire la taille de notre dataset initial de 1,6 million de tweets (800 000 positifs et 800 000 négatifs). Nous avons sélectionné 150 000 instances pour chaque classe, soit un total de 300 000 instances, ce qui a parfaitement répondu à nos exigences et nous a permis d'obtenir un temps d'exécution raisonnable.

IX. Observation de l'environnement Spark

L'environnement Spark utilise des outils de monitoring tels que Spark Web UI, Ganglia et Grafana. Ces outils permettent de suivre en temps réel les performances de Spark, d'analyser les tâches en cours d'exécution, de surveiller les nœuds du cluster et de détecter les goulots d'étranglement éventuels.

Spark Environment :

APACHE

spark

3.3.2

JobsStagesStorageEnvironmentExecutorsSQL / DataFrame

Projet_BDM application UI

Environment

Runtime Information

Name	Value
Java Home	C:\java\jdk
Java Version	19.0.2 (Oracle Corporation)
Scala Version	version 2.12.15

Spark Properties

Name	Value
spark.app.id	local-1681475379010
spark.app.name	Projet_BDM
spark.app.startTime	1681475373625
spark.app.submitTime	1681475373037
spark.driver.extraJavaOptions	-XX:+IgnoreUnrecognizedVMOptions --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.lang.invoke=ALL-UNNAMED --add-opens=java.base/java.lang.reflect=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.net=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.base/java.util.concurrent.atomic=ALL-UNNAMED --add-opens=java.base/java.util.concurrent.locks=ALL-UNNAMED --add-opens=java.base/sun.nio.cs=ALL-UNNAMED --add-opens=java.base/sun.security.action=ALL-UNNAMED --add-opens=java.base/sun.util.calendar=ALL-UNNAMED --add-opens=java.security.jgss/sun.security.krb5=ALL-UNNAMED
spark.driver.host	DESKTOP-03SULAJ
spark.driver.memory	5g
spark.driver.port	55607
spark.executor.cores	10
spark.executor.cores	10
spark.executor.extraJavaOptions	-XX:+IgnoreUnrecognizedVMOptions --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.lang.invoke=ALL-UNNAMED --add-opens=java.base/java.lang.reflect=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.net=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.base/java.util.concurrent.atomic=ALL-UNNAMED --add-opens=java.base/java.util.concurrent.locks=ALL-UNNAMED --add-opens=java.base/sun.nio.cs=ALL-UNNAMED --add-opens=java.base/sun.security.action=ALL-UNNAMED --add-opens=java.base/sun.util.calendar=ALL-UNNAMED --add-opens=java.security.jgss/sun.security.krb5=ALL-UNNAMED
spark.executor.id	driver
spark.executor.memory	5g
spark.master	local[*]
spark.rdd.compress	True
spark.scheduler.mode	FIFO
spark.serializer.objectStreamReset	100
spark.submit.deployMode	client
spark.submit.pyFiles	
spark.ui.showConsoleProgress	true

Resource Profiles

Resource Profile Id	Resource Profile Contents
0	Executor Reqs: cores: [amount: 10] memory: [amount: 5120] offHeap: [amount: 0] Task Reqs: cpus: [amount: 1, 0]

Hadoop Properties

System Properties

Classpath Entries

Spark Jobs :

Completed Jobs (27)

Page: 1

1 Pages, Jump to 1, Show 100 items in a page, Go

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
30	count at DirectMethodHandleAccessor.java:104 count at DirectMethodHandleAccessor.java:104	2023/04/14 13:42:15	34 ms	1/1 (1 skipped)	1/1 (4 skipped)
29	count at DirectMethodHandleAccessor.java:104 count at DirectMethodHandleAccessor.java:104	2023/04/14 13:42:14	1 s	1/1	4/4
28	showString at DirectMethodHandleAccessor.java:104 showString at DirectMethodHandleAccessor.java:104	2023/04/14 13:42:13	74 ms	1/1 (1 skipped)	1/1 (4 skipped)
27	showString at DirectMethodHandleAccessor.java:104 showString at DirectMethodHandleAccessor.java:104	2023/04/14 13:42:10	3 s	1/1	4/4
26	showString at DirectMethodHandleAccessor.java:104 showString at DirectMethodHandleAccessor.java:104	2023/04/14 13:42:02	4 s	1/1	3/3
25	showString at DirectMethodHandleAccessor.java:104 showString at DirectMethodHandleAccessor.java:104	2023/04/14 13:41:59	3 s	1/1	1/1
24	showString at DirectMethodHandleAccessor.java:104 showString at DirectMethodHandleAccessor.java:104	2023/04/14 13:41:59	55 ms	1/1 (1 skipped)	1/1 (4 skipped)
23	showString at DirectMethodHandleAccessor.java:104 showString at DirectMethodHandleAccessor.java:104	2023/04/14 13:41:55	4 s	1/1	4/4
22	count at DirectMethodHandleAccessor.java:104 count at DirectMethodHandleAccessor.java:104	2023/04/14 13:41:54	0.1 s	1/1 (1 skipped)	1/1 (4 skipped)
21	count at DirectMethodHandleAccessor.java:104 count at DirectMethodHandleAccessor.java:104	2023/04/14 13:41:53	1 s	1/1	4/4
20	showString at DirectMethodHandleAccessor.java:104 showString at DirectMethodHandleAccessor.java:104	2023/04/14 13:41:53	60 ms	1/1	1/1
19	csv at DirectMethodHandleAccessor.java:104 csv at DirectMethodHandleAccessor.java:104	2023/04/14 13:41:47	5 s	1/1	4/4
18	csv at DirectMethodHandleAccessor.java:104 csv at DirectMethodHandleAccessor.java:104	2023/04/14 13:41:47	38 ms	1/1	1/1
16	count at DirectMethodHandleAccessor.java:104 count at DirectMethodHandleAccessor.java:104	2023/04/14 13:36:17	34 ms	1/1 (3 skipped)	1/1 (10 skipped)
15	count at DirectMethodHandleAccessor.java:104	2023/04/14 13:36:09	9 s	3/3	10/10

7	showString at DirectMethodHandleAccessor.java:104 showString at DirectMethodHandleAccessor.java:104	2023/04/14 13:32:09	4 s	1/1	<div><div></div></div> 4/4
5	javaToPython at DirectMethodHandleAccessor.java:104 javaToPython at DirectMethodHandleAccessor.java:104	2023/04/14 13:31:15	6 s	1/1	<div><div></div></div> 4/4
4	count at DirectMethodHandleAccessor.java:104 count at DirectMethodHandleAccessor.java:104	2023/04/14 13:31:14	0.3 s	1/1 (1 skipped)	<div><div></div></div> 1/1 (1 skipped)
3	count at DirectMethodHandleAccessor.java:104 count at DirectMethodHandleAccessor.java:104	2023/04/14 13:31:12	2 s	1/1	<div><div></div></div> 4/4
2	showString at DirectMethodHandleAccessor.java:104 showString at DirectMethodHandleAccessor.java:104	2023/04/14 13:31:11	0.3 s	1/1	<div><div></div></div> 1/1
1	csv at DirectMethodHandleAccessor.java:104 csv at DirectMethodHandleAccessor.java:104	2023/04/14 13:31:02	7 s	1/1	<div><div></div></div> 4/4
0	csv at DirectMethodHandleAccessor.java:104 csv at DirectMethodHandleAccessor.java:104	2023/04/14 13:31:00	1 s	1/1	<div><div></div></div> 1/1

Page: 1

1 Pages. Jump to 1 . Show 100 items in a page. Go

▼ Failed Jobs (5)

Page: 1


1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id *	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
31	collect at Word2Vec.scala:191 collect at Word2Vec.scala:191	2023/04/14 13:42:16	11 s	0/1 (1 failed) (1 skipped)	0/4 (1 failed) (4 skipped) (3 killed: Stage cancelled)
17	collect at Word2Vec.scala:191 collect at Word2Vec.scala:191	2023/04/14 13:36:36	10 s	0/2 (2 failed) (2 skipped)	0/8 (1 failed) (4 skipped) (4 killed: Stage cancelled)
12	showString at DirectMethodHandleAccessor.java:104 showString at DirectMethodHandleAccessor.java:104	2023/04/14 13:34:59	10 s	0/1 (1 failed)	0/1 (1 failed)
11	showString at DirectMethodHandleAccessor.java:104 showString at DirectMethodHandleAccessor.java:104	2023/04/14 13:32:43	11 s	0/1 (1 failed)	0/1 (1 failed)
6	collect at C:\Users\pc\AppData\Local\Temp\jvarkit_127323757316572.py:1 collect at C:\Users\pc\AppData\Local\Temp\jvarkit_127323757316572.py:1	2023/04/14 13:31:21	11 s	0/1 (1 failed) (1 skipped)	0/1 (1 failed) (4 skipped)

Page: 1

1 Pages. Jump to . Show items in a page.

Spark Executors :



[Jobs](#)
[Stages](#)
[Environment](#)
[Executors](#)
[SQL / DataFrame](#)

Project_BDM application UI

Executors

[Show Additional Metrics](#)

Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Excluded
Active(1)	0	100.6 KiB / 2.8 GiB	0.0 B	4	0	5	76	88	20 min (1 s)	3.3 GiB	5.5 MiB	5.5 MiB	0
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Total(1)	0	100.6 KiB / 2.8 GiB	0.0 B	4	0	5	76	88	20 min (1 s)	3.3 GiB	5.5 MiB	5.5 MiB	0

Executors

Show entries

Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Thread Dump
driver	DESKTOP-035SULA\S5608	Active	0	100.6 KiB / 2.8 GiB	0.0 B	4	0	5	76	88	20 min (1 s)	3.3 GiB	5.5 MiB	5.5 MiB	Thread Dump

Showing 1 to 1 of 1 entries

[Previous](#)
[Next](#)

X. Conclusion

Ce projet a été une expérience très enrichissante pour nous, car il nous a permis de découvrir et de mettre en pratique différentes techniques d'apprentissage automatique pour l'analyse de sentiments sur Twitter. Nous avons pu travailler avec des outils tels que PySpark et TensorFlow, qui sont très utilisés dans l'industrie pour le traitement de données à grande échelle. Cela nous a permis d'acquérir des compétences importantes dans le domaine du Big Data et du Deep Learning.

Malgré les difficultés rencontrées lors de l'implémentation distribuée de l'algorithme Word2Vec dans PySpark, nous avons pu obtenir des résultats satisfaisants avec une précision de classification de plus de 70% sur le jeu de données de test. Nous avons également eu l'opportunité de travailler avec des données textuelles réelles et de comprendre les particularités de leur traitement.

L'une des techniques d'apprentissage automatique que nous avons utilisées dans ce projet est le Recurrent Neural Network (RNN), qui est un algorithme très important dans le domaine du Deep Learning pour l'analyse de données textuelles. Nous avons pu constater l'efficacité de cette technique pour l'analyse de sentiments sur Twitter

Enfin, nous espérons que notre travail pourra être utile pour d'autres projets de traitement de données textuelles à grande échelle et que nos résultats pourront contribuer à une meilleure compréhension de l'opinion publique

BIBLIOGRAPHIE :

- <https://www.wikipedia.org/>
- <https://www.educative.io/answers/what-are-the-types-of-rnn>
- <https://www.kaggle.com/>
- <https://towardsdatascience.com/recurrent-neural-networks-by-example-in-python-ffd204f99470>
- <https://keras.io/>
- <https://spark.apache.org/>
- <https://stackoverflow.com/>
- <https://blog.octo.com/>
- <https://france.devoteam.com/paroles-dexperts/aller-plus-loin-en-deep-learning-avec-les-reseaux-de-neurones-recurrents-rnns/>
- https://www.cs.toronto.edu/~lczhang/321/lec/rnn_notes.html