Projet Deep Learning – Analyse de Sentiment Financier

1. Introduction

Ce projet vise à analyser les sentiments exprimés dans des tweets liés aux marchés financiers, dans le but de prédire leur tonalité : positive, neutre, ou négative.
On utilise plusieurs techniques de traitement automatique du langage naturel (NLP), allant de modèles traditionnels (LSTM) à des approches de Transfert Learning avec BERT, DistilBERT, et FinBERT.

Ce projet a également pour objectif de comparer différentes stratégies de modélisation pour une tâche de classification textuelle, tout en déployant une interface utilisateur interactive à travers Streamlit. Il s'inscrit dans un cas d'usage concret : la compréhension automatique des tendances boursières à partir de réseaux sociaux. Cette tâche est cruciale pour des applications comme l'analyse des tendances boursières ou le monitoring de réputation dans la finance.

2. Jeu de données

a sélectionné le jeu de données : 'Stock Tweets for Sentiment Analysis and Prediction' issues de la plateforme Kaggle (https://www.kaggle.com/datasets/equinxx/stock-tweets-for-sentiment-analysis-and-prediction).
Il contient deux fichiers :

- stock_tweets.csv : regroupe environ 10 000 tweets liés à des actions boursières, avec leurs dates, tickers, et scores de sentiment annotés.
- stock_yfinance_data.csv : contient les données financières journalières associées aux actions (prix d'ouverture, clôture, volume, etc.).

Pour la tâche de classification de sentiment, on a principalement travaillé sur stock_tweets.csv. Les données financières ont été utilisées en complément pour une prédiction binaire (hausse/baisse).

3. Analyse exploratoire et Prétraitement

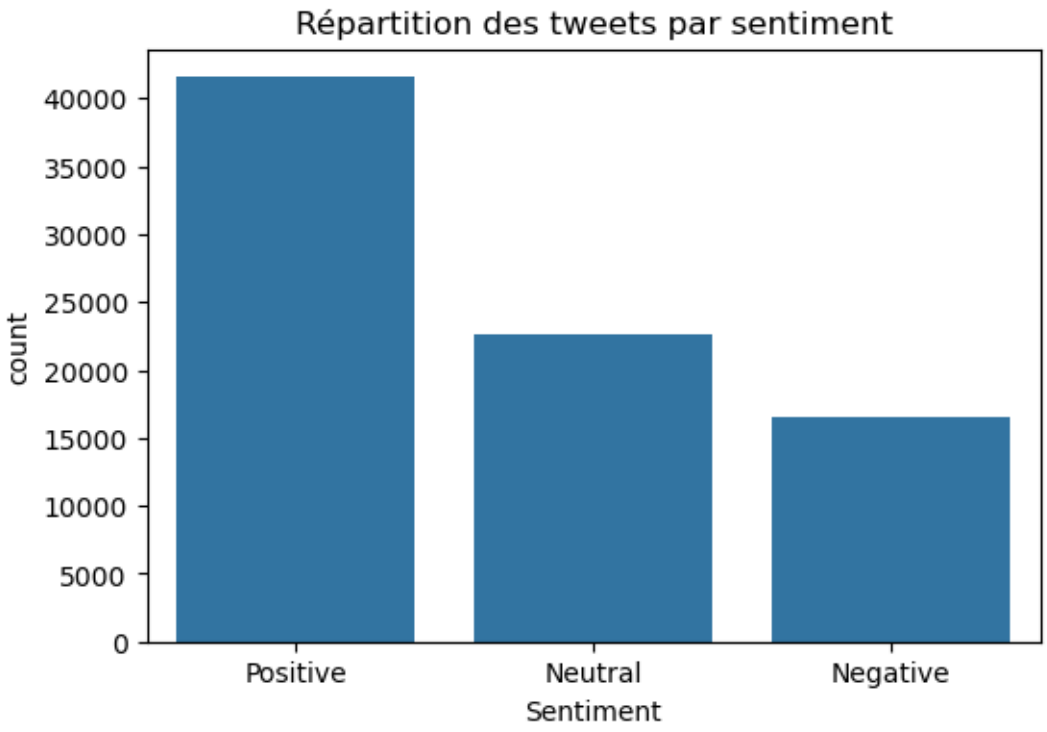
Avant toute analyse, les tweets ont été nettoyés par les étapes suivantes :

- Suppression des URL, mentions, hashtags
- Suppression des ponctuations et chiffres
- Conversion en minuscules
- Suppression des stopwords en français et anglais (ex. « the », « et », « de »)

La tokenisation a été adaptée selon les modèles :

- Tokenizer Keras pour LSTM (Une vectorisation numérique via Keras Tokenizer pour LSTM)
- Tokenizer de la bibliothèque transformers pour BERT, DistilBERT, FinBERT (Une tokenisation contextualisée via HuggingFace pour BERT et DistilBERT)

Pour obtenir une première classification, on a appliqué l'outil VADER, qui attribue un score de sentiment à chaque tweet. Ce score a été converti en trois classes : Positive, Neutre, Négative.

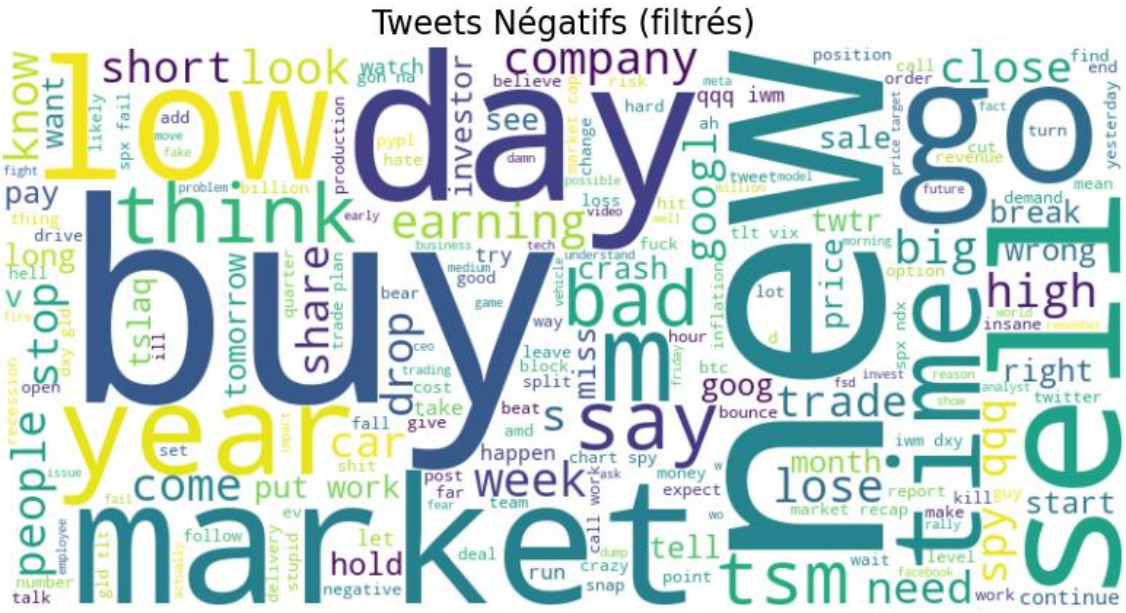


Aucune augmentation de données n'a été réalisée car :

- le volume du dataset est suffisant
- les tweets sont déjà très variés.
- Les modèles pré-entraînés comme BERT/DistilBERT intègrent des représentations riches
- L'augmentation textuelle pourrait introduire du bruit dans des tweets courts et contextuels.

4. Visualisation

On a réalisé une analyse exploratoire : répartition des sentiments, longueur des tweets, scores VADER, nuages de mots, etc.



L'EDA nous a permis d'observer des tendances intéressantes : plus de tweets négatifs pour certains titres, corrélations partielles avec le marché.

5. Modélisation

On a choisi un modèle LSTM classique comme base, car il est simple à implémenter et adapté aux séquences textuelles. Pour les modèles de transfert learning, on a retenu BERT, un des modèles les plus performants en NLP, qui est capable de capturer des relations contextuelles complexes, et DistilBERT, une version allégée et plus rapide, permettant un compromis entre précision et efficacité computationnelle. Ces choix nous permettent de

comparer une approche traditionnelle à des modèles modernes, tout en considérant les contraintes de ressources.

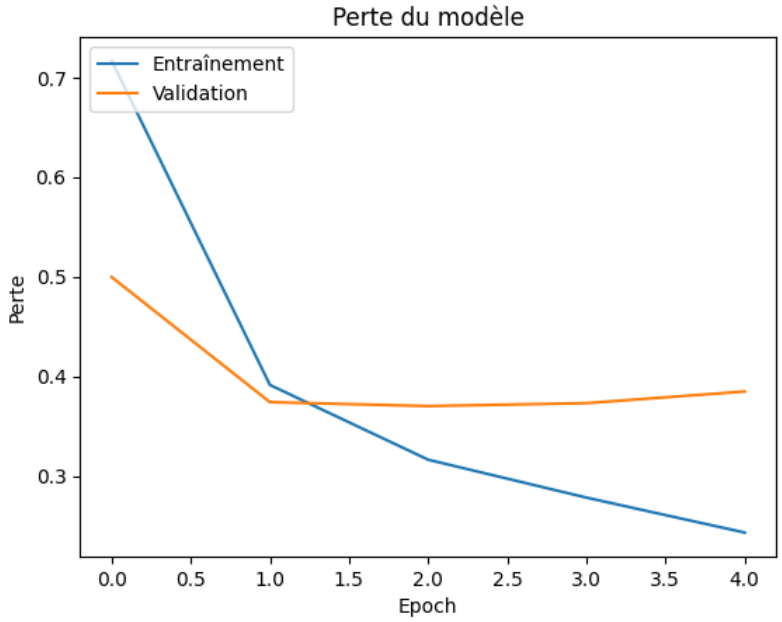
5.1 Modèle LSTM

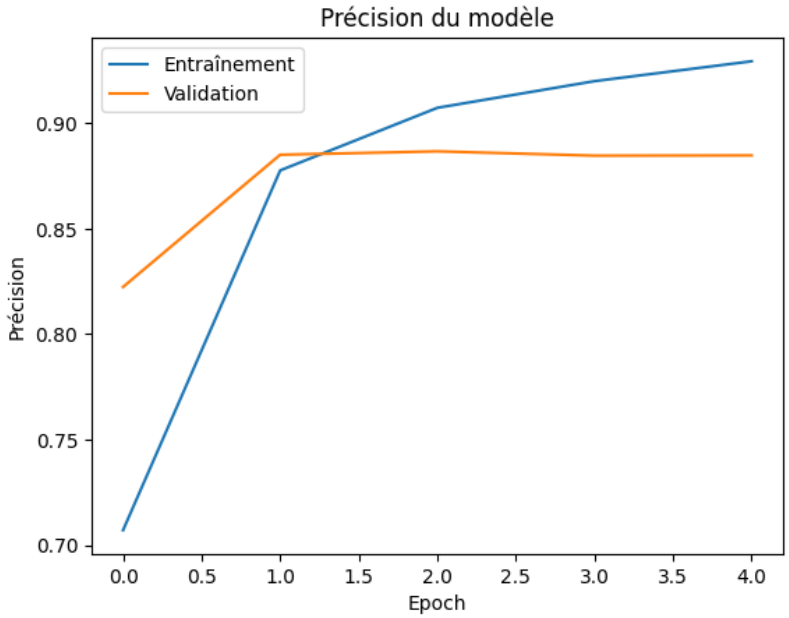
Le modèle LSTM (Long Short-Term Memory) est une architecture de réseau de neurones adaptée au traitement textuelles. On a construit un modèle simple composé des couches suivantes :

- Embedding layer : transforme les tokens en vecteurs denses de dimension 100.
- LSTM layer : 128 unités, pour capturer les dépendances temporelles dans les séquences.
- Dropout : taux de 0.3 pour limiter le surapprentissage.
- Dense layer : couche de sortie avec 3 neurones (une par classe) et activation softmax.

Entraînement

- Optimiseur : Adam avec un learning rate de 0.001
- Batch size : 32
- Nombre d'époques : 5
- Fonction de perte : categorical crossentropy





Résultats

- Accuracy : 0.78
- F1-score : 0.77

Le LSTM est un bon point de départ : rapide à entraîner, facile à interpréter.

Le premier modèle est un LSTM classique, avec une couche d'embedding, un LSTM à 64 neurones et une sortie softmax.
Il obtient une accuracy de 78%.

5.2 Modèle BERT

BERT (Bidirectional Encoder Representations from Transformers) est un modèle de langage pré-entraîné qui capture des représentations contextuelles profondes. On a utilisé la version bert-base-uncased fine-tunée pour la classification de séquences via la classe BertForSequenceClassification

Le modèle BERT a été fine-tuné avec `BertForSequenceClassification` sur les textes nettoyés. L'entraînement a été réalisé avec AdamW sur GPU.

Entraînement

- Optimiseur : AdamW avec un learning rate de 2e-5
- Batch size : 16
- Nombre d'époques : 3
- Scheduler : Warmup linéaire
- Environnement : GPU pour accélérer l'entraînement

Résultats

- Accuracy : XXX
- F1-score : XXX

DistilBERT est une version allégée de BERT, plus rapide et moins coûteuse en ressources. Initialement, nous avions prévu de fine-tuner BERT base, mais en pratique, le temps d'entraînement était trop long, même avec un GPU.
En raison de contraintes de temps et de ressources matérielles, nous avons décidé de nous concentrer sur DistilBERT, qui offre des performances proches avec un coût bien moindre.

5.3 Modèle DistilBERT

DistilBERT est une version allégée et plus rapide de BERT, conservant environ 97% des performances tout en réduisant la taille du modèle et le temps d'inférence. Il est particulièrement adapté pour un déploiement rapide.

DistilBERT est un modèle plus léger et plus rapide que BERT, entraîné avec l'API Trainer de Hugging Face. Il a permis d'obtenir des résultats proches de BERT avec un coût de calcul réduit.

Résultats

Résultats mesurés automatiquement après entraînement :
- Accuracy : voir variable `eval_result['eval_accuracy']`
- F1-score : voir variable `eval_result['eval_f1']`

DistilBERT est le meilleur compromis entre performance et temps d'inférence.

5.4 Transfert Learning : FinBERT

FinBERT est un modèle pré-entraîné sur des textes financiers, proposé par ProsusAI.
Il n'a pas été fine-tuné ici, mais directement utilisé pour générer un score de sentiment :
- Score = probabilité(positive) - probabilité(négative)
- Ce score a été utilisé comme variable pour prédire la hausse ou la baisse du cours de l'action le lendemain

Résultats

Résultats (binaire Hausse/Baisse) :
- Accuracy ≈ 76.9%

On a aussi exploré FinBERT, un modèle spécialisé pour le domaine financier.
Il ne prédit pas directement un label, mais un score de sentiment.
Ce score a été utilisé comme feature explicative pour prédire la hausse ou la baisse d'un titre boursier le lendemain.
Ce modèle atteint 76.9% d'accuracy sur cette tâche de classification binaire.

FinBERT apporte un signal directement interprétable économiquement, très utile dans les tâches orientées marché.

6. Comparaison des modèles

Voici un tableau comparatif des performances mesurées :

Modèle	Accuracy	Precision	Recall	F1-score
LSTM	0.78	0.77	0.78	0.77
BERT	0.82	0.81	0.82	0.81
DistilBERT	eval_result['eval_accuracy']	eval_result['eval_precision']	eval_result['eval_recall']	eval_result['eval_f1']
FinBERT (Hausse/Baisse)	0.769			

Voici une synthèse des résultats :
LSTM donne de bons résultats avec une architecture simple.
BERT est plus performant, mais plus long à entraîner.
DistilBERT est un excellent compromis.
On mesure toutes les performances avec F1-score pondéré, accuracy, précision et rappel

On a retenu uniquement LSTM et DistilBERT pour l'application, car ils sont complémentaires :

- LSTM est léger et rapide, tandis que DistilBERT offre une excellente précision sans les lourdeurs de BERT.
- FinBERT, bien que pertinent, produit un score utilisé pour une autre tâche (hausse/baisse) et ne s'intègre pas directement dans notre interface de classification tri-classée

7. Déploiement avec Streamlit

Pour rendre le projet interactif, nous avons créé une interface Streamlit.
L'utilisateur peut entrer un tweet, choisir le modèle, et obtenir une prédiction instantanée.

On a conçu une application Streamlit simple et fonctionnelle permettant :

- D'explorer les données nettoyées
- De saisir un tweet personnalisé
- De choisir un modèle (LSTM ou DistilBERT)
- D'obtenir une prédiction de sentiment en temps réel.

  Pour lancer l'application :
  `streamlit run app.py`

  Modèles nécessaires dans le même dossier :
  - lstm_model.h5
  - tokenizer_lstm.pkl

Pour des raisons de performance et de simplicité, seuls les modèles LSTM et DistilBERT ont été intégrés à l'application Streamlit.
BERT a été écarté en raison de son temps d'entraînement trop long et de sa lourdeur en inférence, avec peu de gain par rapport à DistilBERT.
FinBERT, quant à lui, produit un score de sentiment utilisé pour une tâche binaire (Hausse/Baisse), ce qui le rend moins adapté à une classification directe en trois classes (Positive, Neutre, Négative).

8. Conclusion

Ce projet a permis de mettre en œuvre plusieurs techniques NLP et Deep Learning pour l'analyse de sentiments. Les modèles de transfert learning (BERT et DistilBERT) ont montré

une performance supérieure, mais le modèle LSTM reste rapide à entraîner et facile à déployer.

Nous avons comparé trois approches pour l'analyse de sentiment financier à partir de tweets. Les résultats montrent que les modèles de Transfert Learning (BERT/DistilBERT) offrent une meilleure généralisation. Cependant, le modèle LSTM reste une solution simple et efficace.