Rapport du projet du Groupe 2 : Analyse des Avis Clients

1. Introduction

□ Contexte et objectif du projet

Dans un monde où les entreprises sont de plus en plus tournées vers l'expérience utilisateur, comprendre les avis clients est devenu un enjeu stratégique. Les plateformes comme Amazon, Trustpilot ou IMDB regorgent de commentaires qui peuvent fournir des insights précieux sur la satisfaction des utilisateurs. Cependant, traiter ces données manuellement est une tâche fastidieuse et inefficace.

Ce projet vise à développer un modèle d'analyse de sentiments capable de classifier automatiquement les avis clients en deux catégories : **positif et négatif**. En exploitant les avancées des modèles de traitement du langage naturel (NLP), nous avons testé deux approches : l'extraction de caractéristiques à l'aide d'un transformeur (feature extraction) et l'affinage d'un modèle pré-entraîné (fine-tuning). L'objectif final est d'intégrer ce modèle dans une API interactive permettant aux entreprises d'exploiter ces analyses de manière efficace.

2. Architecture du projet

□ Présentation des données (source, prétraitement)

Les données utilisées proviennent de sources publiques contenant des avis clients. Nous avons sélectionné le dataset d'IMDB. Ce jeu de données contient des commentaires associés à des labels indiquant leur sentiment (positive, négative).

□ Étapes de prétraitement des données :

1. **Nettoyage des données** : Suppression des caractères spéciaux, des balises HTML et des stop words.
2. **Tokenisation** : Conversion des phrases en tokens exploitables par les modèles NLP.
3. **Encodage des labels** : Transformation des sentiments en valeurs numériques (positive = 1, négative = 0).
4. **Découpage des données** : Séparation en ensembles d'entraînement et de validation pour évaluer les performances du modèle.

□ Modèle NLP utilisé (DistilBERT, regression logistique, etc.)

Pour ce projet, nous avons testé deux approches :

- **Extraction de caractéristiques avec un transformeur** : Utilisation de **DistilBERT** comme extracteur de features et entraînement d'un modèle de **régression logistique** pour la classification.
- **Fine-tuning d'un modèle pré-entraîné** : Ajustement des poids de **DistilBERT** pour affiner les performances sur notre tâche spécifique.

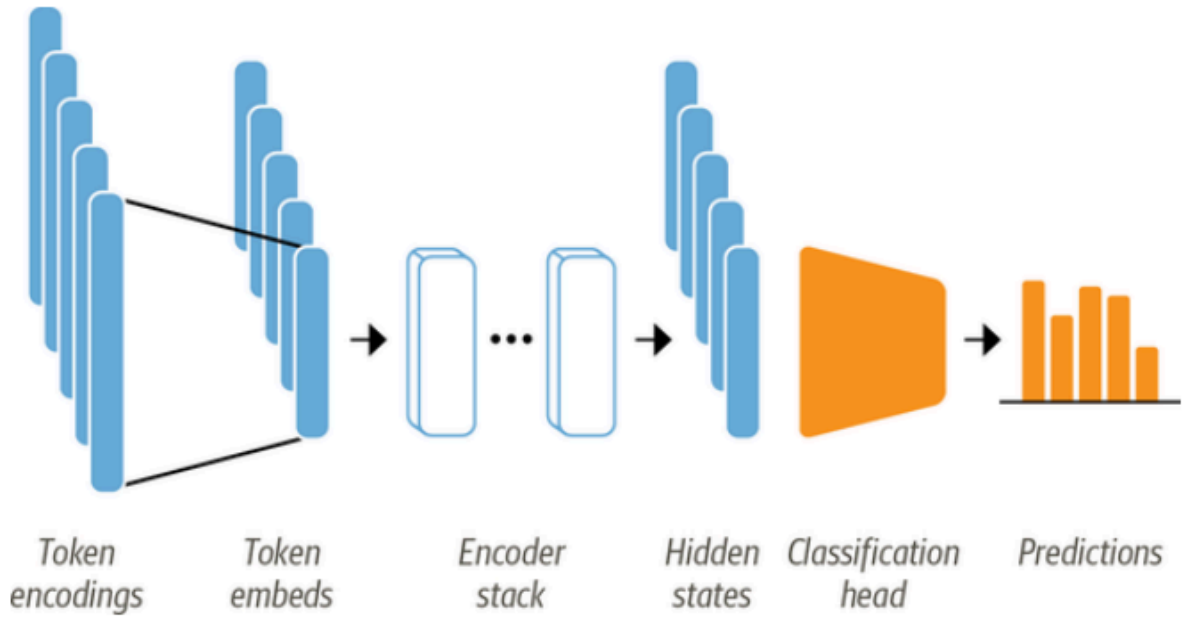
❖ Approche adoptée (Feature extraction vs Fine-tuning)

- **Feature extraction** : Cette méthode consiste à utiliser DistilBERT pour générer des représentations des avis, qui sont ensuite passées à un modèle de classification classique comme la régression logistique. L'avantage est une rapidité d'entraînement et une moindre consommation de ressources, car les poids du transformeur restent figés.
- **Fine-tuning** : Ici, nous adaptons directement les poids de DistilBERT à notre jeu de données en réentraînant certaines couches du modèle.

3. Explication du modèle NLP utilisé

Les modèles de type Transformer, comme DistilBERT, sont initialement préentraînés sur de vastes corpus de texte pour prédire des mots masqués dans une phrase. Cependant, pour les adapter à des tâches spécifiques comme la classification de texte, il est nécessaire de modifier légèrement leur architecture.

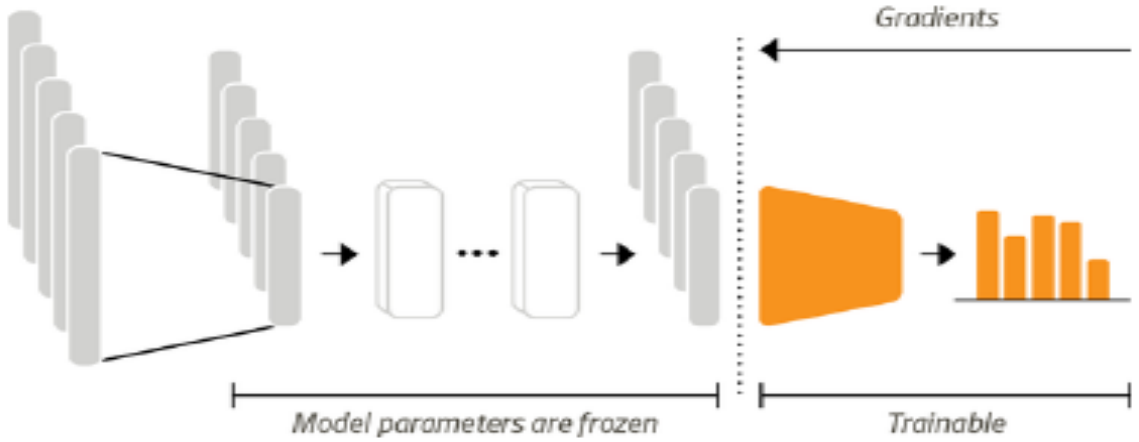
❖ Architecture de DistilBERT pour la classification de texte



L'architecture de DistilBERT suit le schéma général des modèles basés sur un encodeur. Le processus de traitement des textes est le suivant :

1. **Tokenisation** : Le texte est transformé en tokens et représenté sous forme d'encodages one-hot. La dimension de ces encodages est définie par la taille du vocabulaire du tokenizer (entre 20 000 et 200 000 tokens).
2. **Embeddings** : Ces encodages sont ensuite convertis en vecteurs d'embeddings situés dans un espace de dimension inférieure, ce qui permet de capturer les relations sémantiques entre les mots.
3. **Passage dans l'encodeur** : Les embeddings traversent les différentes couches de l'encodeur du modèle. Ce processus génère un état caché pour chaque token, qui capture des informations contextuelles sur le texte.
4. **Classification** : En phase de préentraînement, ces états cachés sont utilisés pour prédire les mots masqués. Pour la classification de texte, cette couche de prédiction est remplacée par une couche de classification spécifique qui permet d'attribuer une étiquette à la séquence complète.

❖ Utilisation de DistilBERT comme extracteur de caractéristiques



Une approche alternative à l'entraînement complet du modèle consiste à utiliser DistilBERT comme **extracteur de caractéristiques** (Feature Extractor). Dans ce cas, les poids des couches de l'encodeur sont gelés, et seuls les états cachés générés par le modèle sont utilisés comme entrée pour un classificateur.

Cette méthode présente plusieurs avantages :

- **Rapidité d'entraînement** : Seul le classificateur est entraîné, ce qui réduit considérablement le temps de calcul.
- **Faible consommation de ressources** : Idéal lorsque l'accès aux GPU est limité, car les états cachés peuvent être pré-calculés et réutilisés.
- **Flexibilité** : Le classificateur peut être une simple couche neuronale ou un algorithme ne nécessitant pas de gradients, comme une forêt aléatoire (Random Forest).

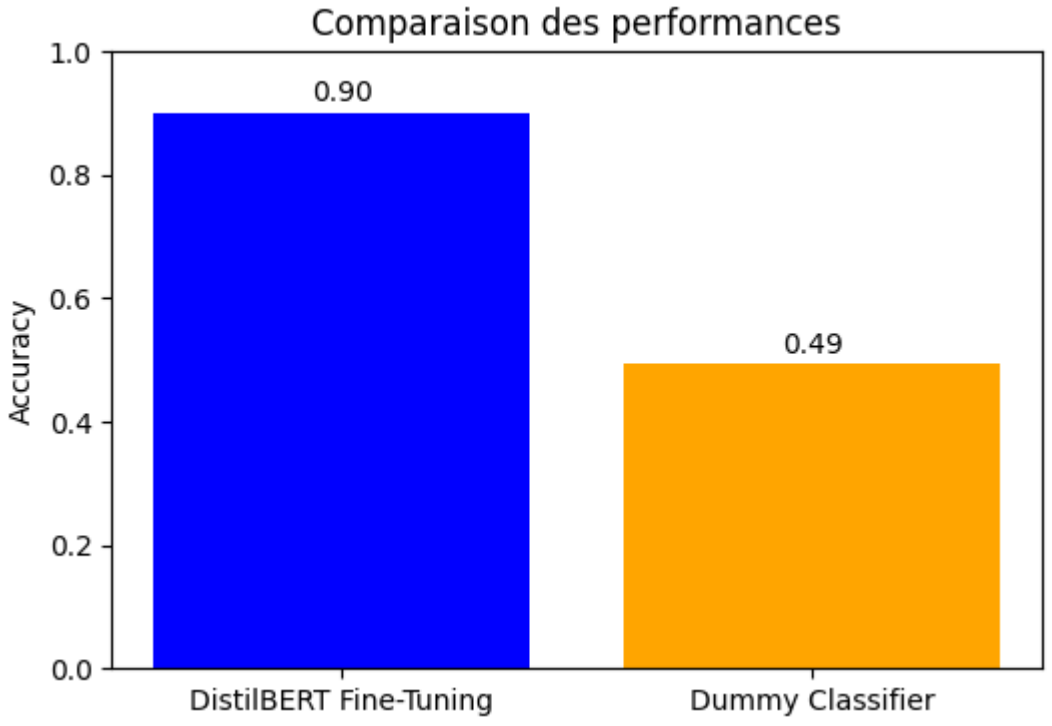
4. Analyse des performances et résultats

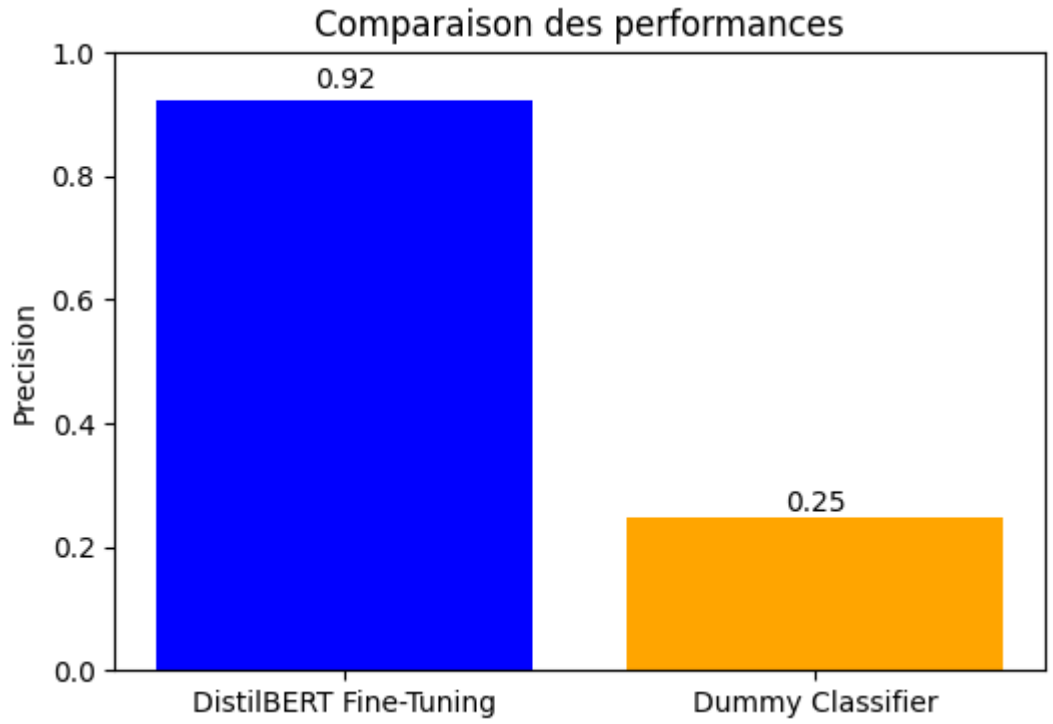
□ Comparaison des approches : Feature Extraction vs Fine-Tuning

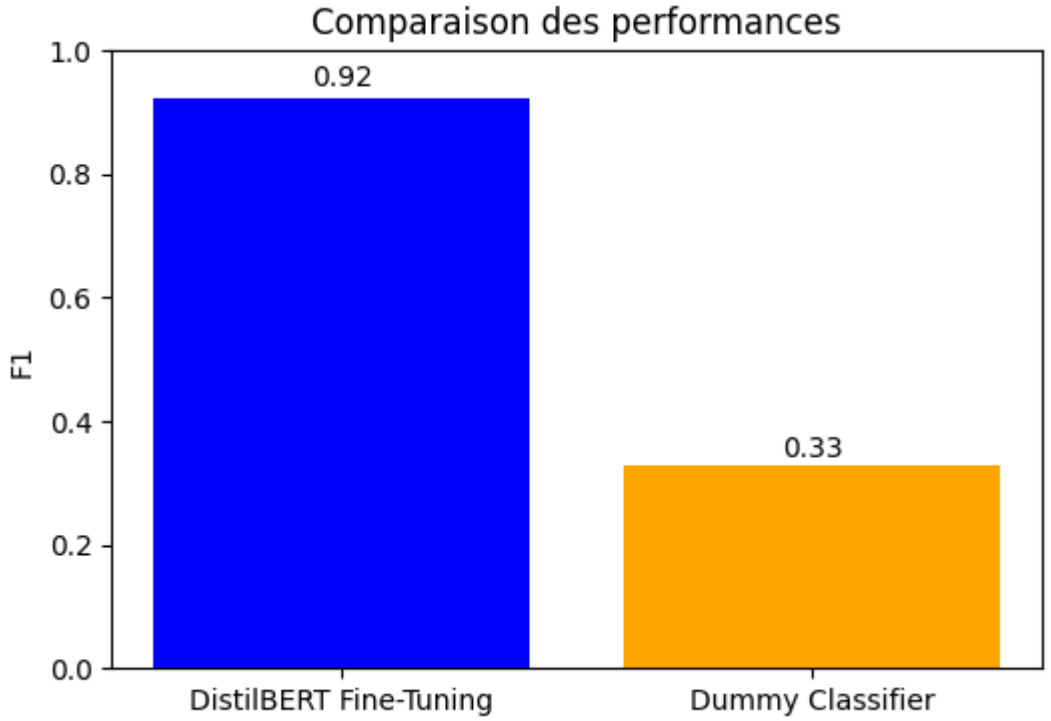
Deux méthodes ont été évaluées :

1. **Fine-Tuning de DistilBERT** :
   - Le modèle entier est ajusté sur les données d'entraînement.
   - Meilleure adaptation aux avis clients spécifiques.
   - Plus coûteux en ressources mais performant.
2. **Dummy Classifier (classificateur de base)** :
   - Basé sur une stratégie naïve (ex. prédictions aléatoires ou majorité).
   - Sert de point de comparaison pour évaluer la valeur ajoutée du modèle.

✔ Métriques d'évaluation







Les graphiques ci-dessus montrent que le fine-tuning de **DistilBERT atteint une précision de 92%**, un **accuracy de 90% avec un F1-score de 92%** tandis que le **Dummy Classifier a une précision de 25%, un accuracy de 49% et un**

F1-score de 33%. Cela confirme l'efficacité de l'approche basée sur un modèle préentraîné adapté à la tâche.

✔ Interprétation

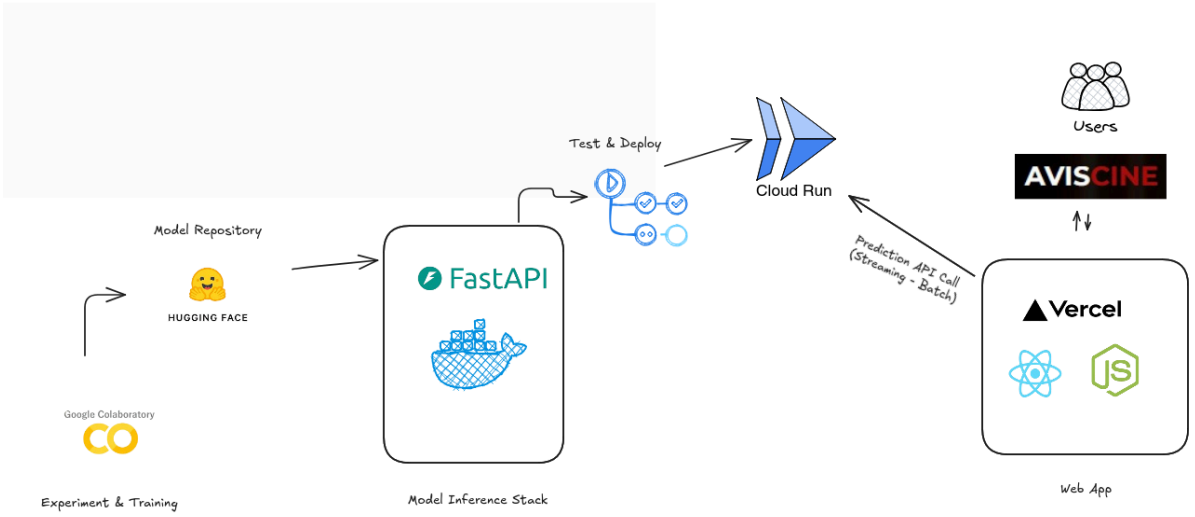
- **DistilBERT fine-tuné est nettement supérieur** : Il apprend les relations entre les mots et comprend mieux le contexte des avis clients, ce qui explique son **fort taux de précision et de F1-score**.
- **Le Dummy Classifier est inefficace** : Il ne repose pas sur une véritable compréhension du texte et ne fait qu'appliquer des règles simples, ce qui explique ses faibles performances.
- **Différence entre Accuracy et F1-score** : L'**accuracy** peut être trompeuse si les classes sont déséquilibrées, mais le **F1-score élevé (92%)** confirme que le modèle fine-tuné fonctionne bien pour toutes les catégories d'avis.

Le fine-tuning de **DistilBERT améliore drastiquement la classification des avis clients**, contrairement au Dummy Classifier qui agit comme un simple point de comparaison.

5. Fonctionnement de l'équipe

La réussite du projet a été du début à la fin rendue possible grâce à la participation active des membres dont le rôle de chacun a été clairement défini dès le départ . Ainsi, la team a pris connaissance du projet et a réglé dans la première semaine la méthodologie concernant la problématique du projet , les données à utiliser et le(s) modèle(s) à entraîner . Ensuite , une partie de l'équipe a exécuté la phase de preprocessing et d'entraînement du modèle sous le suivi des autres qui apportaient leurs remarques et suggestions . Grâce à la pluridisciplinarité des membres , très vite , les autres membres se consacraient à la conception de l'API , au déploiement des assets et à la conception de l'interface web dans une symbiose tant bien que mal . La rédaction du rapport et des autres éléments de présentation a été aussi fluide due au travail d'analyse et de planification effectué dès le début de l'aventure.

6- Déploiement de l'application



Le modèle est déployé sous forme d'application web communicant avec l'API de prédiction FastAPI . Dans un premier temps, l'entraînement et l'optimisation du modèle dont l'artefact final est envoyé vers un dépôt de modèle sur Hugging Face . Le modèle est utilisé par le code de l'API d'inférence FastAPI hébergé sur Github et déployé sous forme de container sur Google Cloud Run après validation des tests unitaires après chaque commit de code . Google Cloud Run est un service de Google Cloud qui permet de déployer facilement les applications conteneurisées sur le web . L'API est servie à l'application web dénommée AvisCINE directement accessible aux utilisateurs finaux . Ce flux continue permet de garantir une application stable et évolutive.

7. Perspectives

Bien que le projet semble achevé , nous restons convaincus que des points sont améliorés notamment :

- *L'automatisation du processus de collecte de données jusqu'à l'entraînement du modèle dans une pipeline*

- *L'optimisation du modèle pour des performances optimales en production*

- *Cycle de vie évolutif et complet (Data Collection - Preprocessing -Training - Evaluation - Inference - Serving)*

8. Conclusion

Ce projet a permis de développer un modèle de classification des avis clients en utilisant des techniques avancées de NLP. Différentes approches ont été comparées, montrant l'efficacité du fine-tuning par rapport à l'extraction de caractéristiques.

Le modèle entraîné sera intégré dans une API interactive permettant aux entreprises d'exploiter ces analyses de manière efficace.