Miali Matteo 20308845

IFT 3700 - HW6
Finetuning

1. Distillbert pour Classification de Sentiments

Modèle de Base

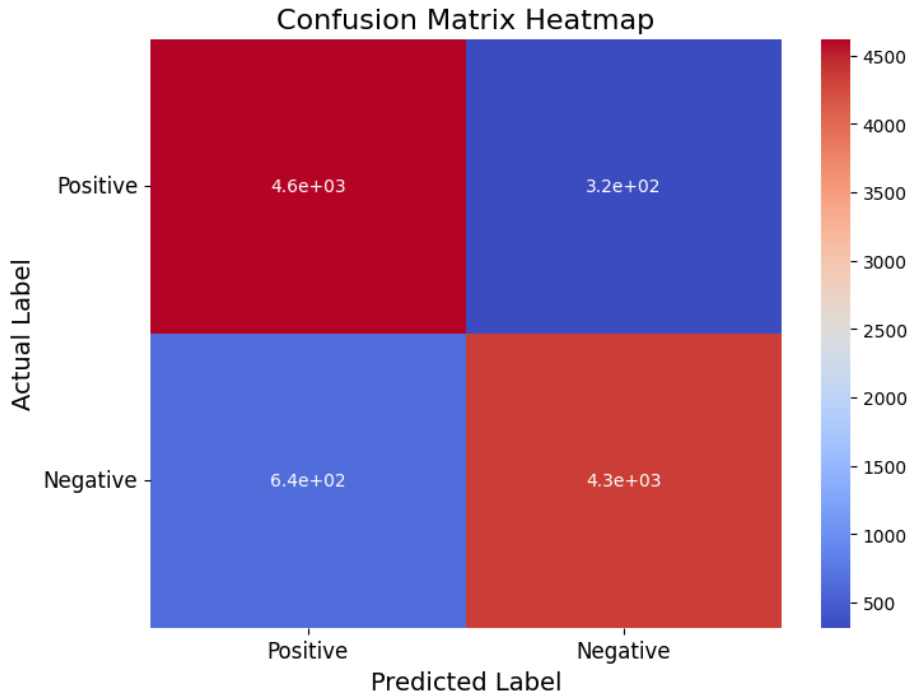
Le modèle initial utilisé est **DistilBERT** pour la classification binaire, configuré avec les **Hyperparamètres** suivants :

- learning_rate=2e-5,
- per_device_train_batch_size=16,
- per_device_eval_batch_size=16,
- num_train_epochs=1,
- weight_decay=0.01)

Les résultats initiaux de l'évaluation sont les suivants :

- **Exactitude** : 90.36%
- **Précision** : 93.19%
- **Rappel** : 87.16%
- **Score F1** : 90.08%

HeatMap :



Méthode 1 : Ajustement du Taux d'Apprentissage et du Nombre d'Époques

Le taux d'apprentissage et le nombre d'époques sont des hyperparamètres qui influencent la vitesse et la convergence du modèle. Un taux d'apprentissage trop élevé peut entraîner une convergence instable, tandis qu'un taux trop bas peut ralentir l'entraînement. Un nombre d'époques insuffisant peut provoquer du sous-apprentissage, tandis qu'un grand nombre peut conduire à un surapprentissage.

Ajustements Apportés

- **Taux d'Apprentissage** : Ajusté de 2e-5 à 3e-5.
- **Nombre d'Époques** : Augmenté de 1 à 3.

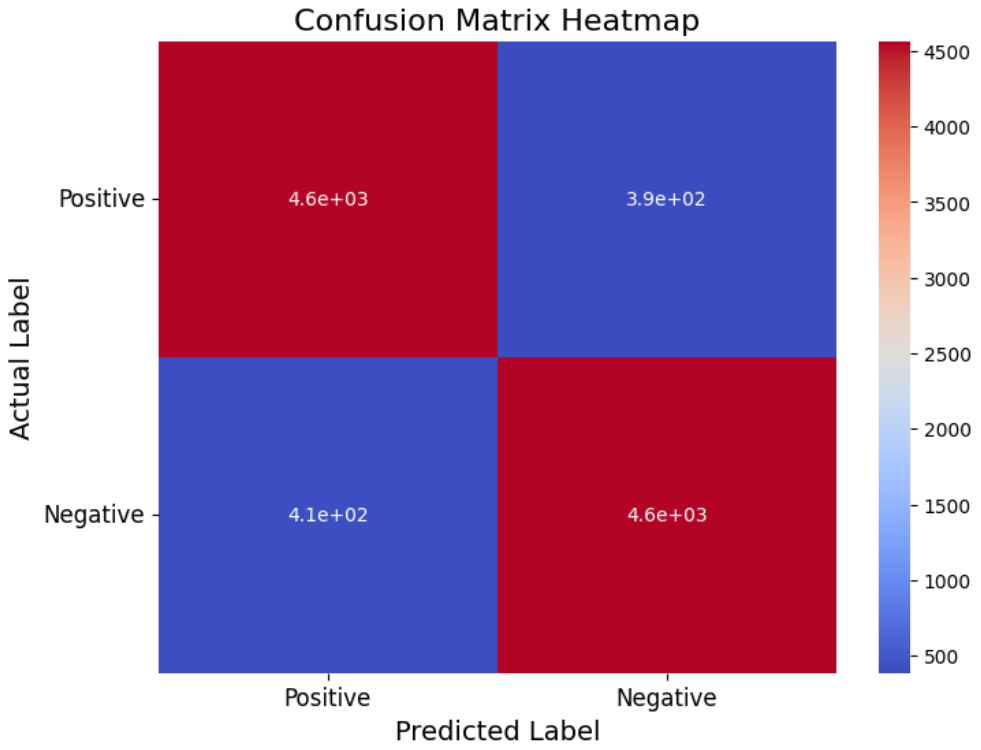
Un taux d'apprentissage plus élevé pourrait accélérer la convergence, mais risque également d'entraîner une oscillation autour du minimum ou même une divergence. Un taux de 3e-5 est un compromis, suffisamment élevé pour accélérer l'apprentissage tout en maintenant la stabilité observée avec 2e-5 précédemment.

Une seule époque peut être insuffisante pour que le modèle capture les nuances des données. En augmentant à 3 époques, le modèle a plus d'opportunités d'ajuster ses poids et d'améliorer sa compréhension des patterns dans les données.

Un nombre plus élevé d'époques peut conduire au surapprentissage, la sélection de 3 époques représente une augmentation modérée qui, combinée avec une décroissance de poids appropriée (weight_decay=0.01), aide à équilibrer l'apprentissage et la généralisation du modèle.

Les résultats de l'évaluation sont les suivants :

- **Exactitude** : 91.91%
- **Précision** : 92.16%
- **Rappel** : 91.68%
- **Score F1** : 91.92%
  HeatMap :



Méthode 2 : Ajustement de la Décroissance de Poids et de la Taille du Batch

La décroissance de poids aide à régulariser le modèle en évitant des poids excessivement grands, ce qui peut réduire le surapprentissage. La taille du batch influence la stabilité des gradients et la vitesse de convergence. Une taille de batch appropriée peut améliorer l'efficacité de l'entraînement.

Ajustements Apportés

- **Décroissance de Poids** : Ajustée de 0.01 à 0.05.
- **Taille du Batch** : Augmentée de 16 à 32.

Un taux de décroissance de poids trop élevé peut peut provoquer du sous apprentissage en contraignant excessivement les poids, tandis qu'un taux trop bas peut ne pas suffire à prévenir le surapprentissage. Un ajustement à 0.05 est un bon compromis, empêchant le sur ajustement sans compromettre la capacité du modèle à apprendre des représentations complexes.

Une taille de batch plus grande permet une estimation plus stable et précise des gradients lors de la rétropropagation. En augmentant la taille du batch, la

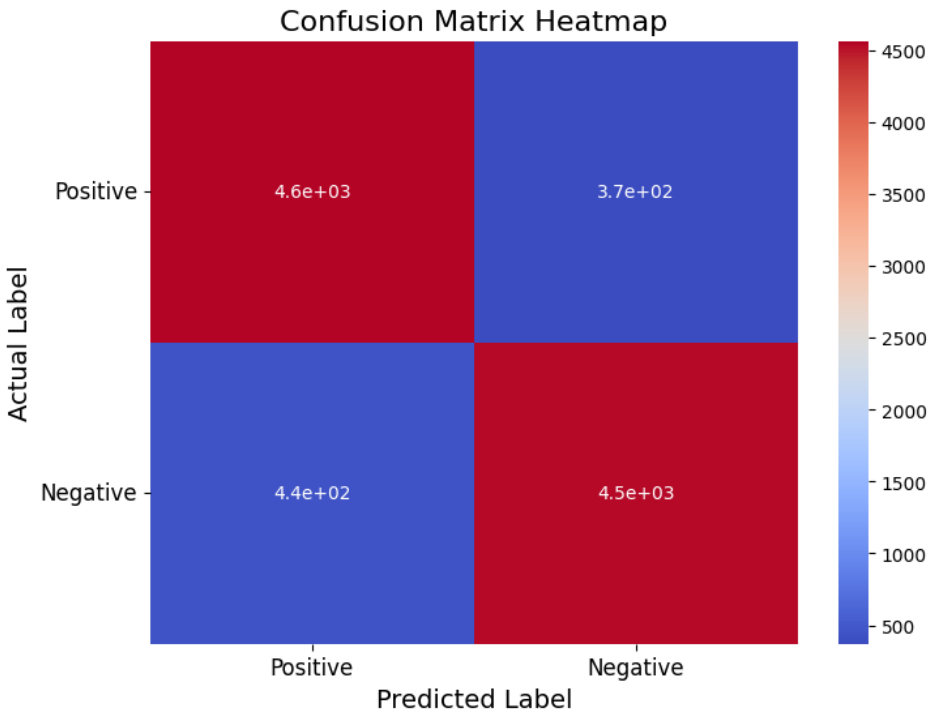
variance des gradients diminue, ce qui peut faciliter l'apprentissage, surtout dans les premières phases d'entraînement.

Une taille de batch de 32 est un bon compromis entre efficacité computationnelle et consommation mémoire.

Les résultats de l'évaluation sont les suivants :

- **Exactitude** : 91.84%
- **Précision** : 92.44%
- **Rappel** : 91.20%
- **Score F1** : 91.81%

HeatMap :



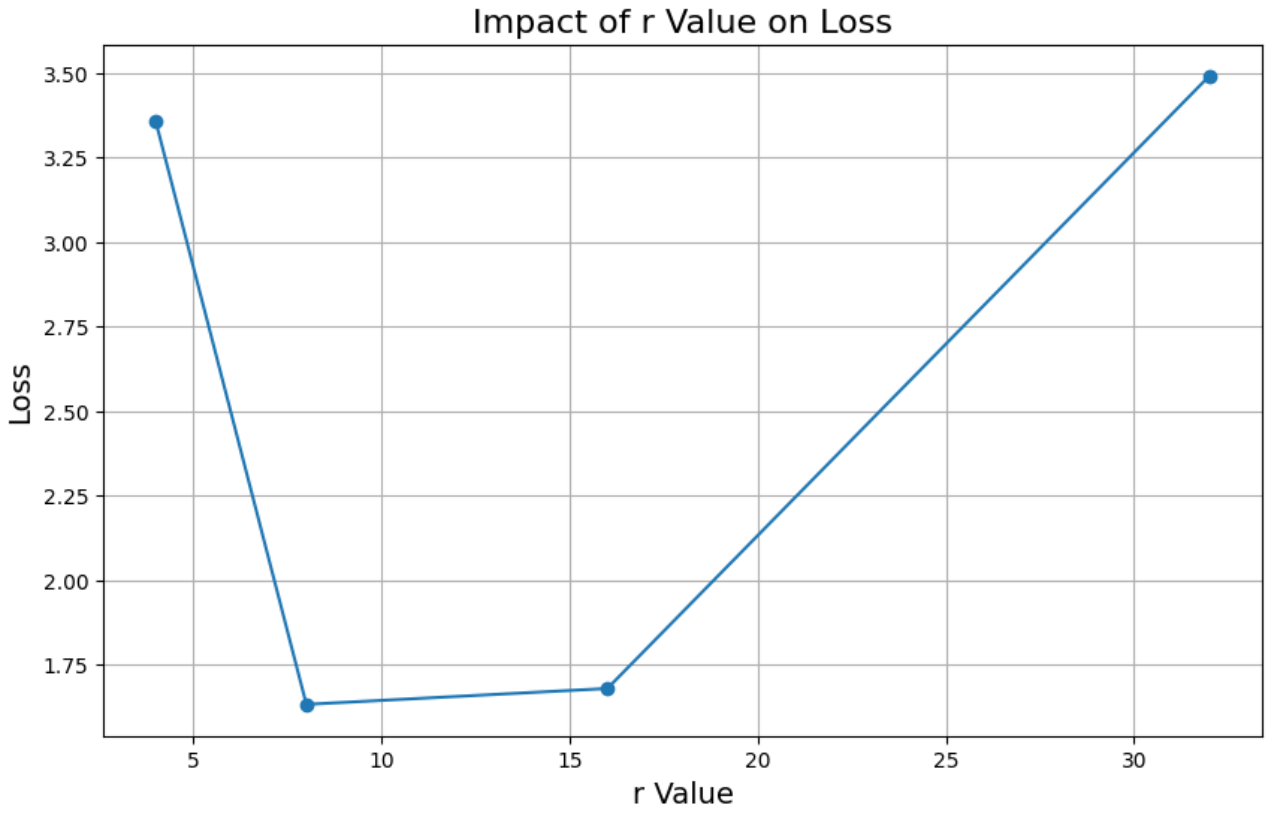
Comparaison entre Méthode 1 et Méthode 2:

Méthode	Exactitude	Précision	Rappel	Score F1
Méthode 1	91.91%	92.16%	91.68 %	91.92%
Méthode 2	91.84%	92.44%	91.20 %	91.81%

2. Finetune d’OpenLlama-2-3b

Une valeur élevée de r permet au modèle d'apprendre des représentations plus complexes, ce qui peut potentiellement améliorer les performances sur des tâches délicates.

Donc augmenter r devrait réduire la perte jusqu'à un certain point, après quoi les gains pourraient diminuer ou même se dégrader en raison du sur-ajustement ou des contraintes de ressources.



On voit bien sur le plot que à une LORA r-Value de 4 le modele est clairement sous-entrainer et que à une r-Value de 32 le modele est sur-entrainer.
On peut supposer avec ces quatre échantillons que l’optimum de complexité du modèle doit être entre un LORA r value de 8 et de 16.

Code pour ajouter un exemple au jeu de données:

```
exemple = {
    'input': "la question",
    'output': "la reponse"
}

dataset = dataset.add_item({'input': exemple['input'], 'output': exemple['output']})
```