INF8225 - Analyse de Sentiment avec Transformers

WEISBECKER Lisa [1] & BENDELAC Elie[2]
Polytechnique Montréal
{lisa.weisbecker, elie.bendelac}@polymtl.ca

April 29, 2024

Abstract

Le domaine en expansion de l'analyse de sentiments exploite les capacités profondes des grands modèles de langage (LLMs) pour interpréter et classifier les retours d'utilisateurs textuels. Dans notre projet, nous proposons une analyse comparative de deux modèles Transformers prédominants, BERT et GPT, en employant des techniques de fine-tuning pour optimiser leurs performances pour des tâches d'analyse de sentiments. Plus précisément, la recherche examine l'impact de divers hyperparamètres tels que le nombre de couches gelées, les couches additionnelles, les taux d'apprentissage, et les époques d'entraînement sur l'efficacité des modèles. Utilisant des ensembles de données tels que les critiques de films IMDb pour l'entraînement et les avis Yelp pour l'évaluation, ce projet explore non seulement l'amélioration de la précision des prédictions mais aussi la prévention du surentraînement et de l'oubli catastrophique, deux défis critiques dans le finetuning des réseaux neuronaux. Les résultats expérimentaux montrent que les stratégies de finetuning telles que le gel des couches et l'ajout de couches entraînables renforcent significativement la précision et la transférabilité des modèles sans succomber au surentraînement ou à l'oubli des informations précédemment apprises. À travers des tests rigoureux, cette étude fournit des insights sur les dynamiques subtiles de l'entraînement des LLMs, offrant une comparaison détaillée de l'adaptabilité des modèles à l'analyse de sentiments, avec un focus particulier sur le maintien de la robustesse à travers différents ensembles de données. Les résultats soulignent le potentiel des approches de finetuning sur mesure pour créer des modèles non seulement précis mais aussi polyvalents dans les applications du monde réel.

1 Introduction

Obtenir un retour des usagers est fondamental. À la fois pour le pourvoyeur d'un produit ou service, qui pourra alors s'adapter à sa clientèle, et pour les consommateurs, qui vont pouvoir choisir un film, un restaurant ou un réfrigérateur en fonction des avis données par d'autres personnes ayant les mêmes envies ou besoins. De nombreuses plateformes existent permettant de déposer son avis ou de consulter celui des autres, comme yelp ou google maps. Mais ces immenses bases de données, créées et complétées régulièrement par les usagers, doivent être traitées, et cela peut s'avérer long et fastidieux pour une personne seule. En 2002, plusieurs projets de recherches prennent pour objectif d'analyser les sentiments des avis textuels grâce à l'intelligence artificielle. Bo Pang et al. [Pang *et al.*, 2002], par exemple, ont été les premiers à chercher à séparer les avis en deux groupes, positifs et négatifs.
Le développement récent du deep learning et des larges modèles de language a offert de nouvelles possibilités de recherches quant à l'analyse de sentiments. Plusieurs articles suivent cette voie, comme celui de Krugmann et Hartmann en 2024 [Krugmann and Hartmann, 2024], qui montre la puissance de l'intelligence artificielle générative et notamment des larges modèles de langages (LLMs) pour l'exécution de cette tâche. Un puissant outil accompagnant ces modèle est le finetuning : L'entrainement sur une base de données spécifique et réduite d'un modèle pré-entrainé pour le rendre plus apte à exécuter une tâche bien particulière. Cette procédure est décrite et testé dans de nombreux articles, dont celui de Nguyen et al., en 2020 [Nguyen *et al.*, 2020].

Dans le cadre de ce projet, deux LLMs particulier, GPT et Bert ont été étudié. Plusieurs méthodes ont été développés sur ces modèles pour obtenir le meilleur finetuning possible. Entre autres : l'ajout de nouveaux paramètres, comme montré par Raschka, S. [Raschka, 2023] en 2023, ainsi que le gel de certaines couches du modèle pré-entrainé, explicité par Takyar et al [Takyar, 2023]. Ce projet a pour but de comparer les performances de ces deux LLMs une fois un finetuning orienté vers l'analyse de sentiments réalisé. Il a également pour but de tester et comparer l'efficacité de différents ensembles d'hyperparamètres de sorte à obtenir le modèle le plus efficace possible.
Il ne s'agit pas seulement de maximiser le pourcentage de prédiction correcte du modèle en minimisant la perte. Un premier obstacle classique de l'intelligence artificielle est le surentrainement. Ce qu'il faut absolument éviter si l'on veut que le modèle soit transférable à d'autres ensembles de données. Un autre obstacle très important du finetuning est l'oubli catastrophique. C'est un scénario ou l'impact du finetuning est trop grand et où le modèle "oubli" le pré-entrainement. Or, ce dernier a été effectué sur un ensemble bien plus grand et plus représentatif, il est nécessaire d'en conserver un maximum si l'on veut éviter de nombreux

biais. Les méthodes citées précédemment ont pour but de maximiser la prédiction tout en minimisant l'oubli catastrophique. Pour le vérifier, une des méthodes les plus simple est de tester le modèle sur un autre ensemble de données précédemment traitable, mais distinct de la tache concernée par le finetuning, comme présenté dans l'article de Luo et al., en 2023 [Luo *et al.*, 2023]. L'oubli catastrophique peut alors être quantifié par la baisse de qualité de la prédiction.

2 Approche théorique et méthodologie du projet.

2.1 BERT

L'impact de différents hyperparamètres sera testé sur le finetuning d'un modèle pré-entrainé de Bert. L'objectif étant d'obtenir les meilleurs résultats possibles tout en évitant le surentrainement et en minimisant l'oubli catastrophique. La procédure de finetuning de base été réalisé grâce à tensorflow en se basant sur l'article de Moonat, D. [Moonat, 2021].

2.1.1 Les différents ensemble de données

L'entrainement du modèle s'est fait à partir de l'ensemble de donnés "imdb_reviews" [TensorFlowDatasets, 2024] séparé en un ensemble d'entrainement et un ensemble de validation.

Nous avons testé le surentrainement et la transférabilité sur un premier ensemble de test "yelp_polarity_reviews" [TensorFlowDatasets, 2024]. C'est un ensemble différent, mais proche et correspondant à la même tâche que l'ensemble d'entrainement : Classer différents avis entre positifs et négatifs.

L'oubli catastrophique, c'est-à-dire la perte d'une partie du pré-entrainement dû au finetuning a également été étudié. Pour cela, nous avons, à la suite du finetuning, testé notre modèle sur un second ensemble de test, provenant de l'ensemble "Glue" [TensorFlowDatasets, 2024]. Nous l'avons modifié de sorte qu'il corresponde aux caractéristiques du modèle. C'est-à-dire une chaine de caractère en entrée et deux classes possibles en sortie. Mais la tâche n'a rien à voir avec celle des données précédentes. Ici, Bert doit classer des phrases en anglais selon leur acceptabilité linguistique. Bert ne pourra donc compter que sur ce qui lui reste de son pré-entrainement pour traiter cet ensemble.

2.1.2 Les hyperparamètres étudiés

Le premier hyperparamètre que nous avons étudié est le nombre de couches gelés. Pour cela, un certain nombre de couches (en commençant par les premières) ont été rendu non-entrainable avant le finetuning.

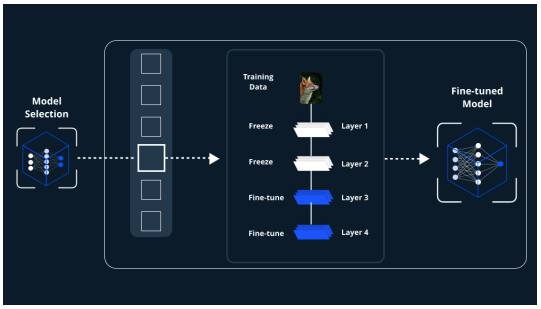


Figure 1: Schéma du Finetuning par gèle de couche [Takyar, 2023]

Le second hyperparamètre étudié correspond au nombre de couches additionnelles. Nous avons, à la suite du modèle pré-entrainé, ajouté une ou deux couches supplémentaires. Chacune d'entre elle correspond à une couche dense de 256 paramètres avec pour fonction d'activation ReLu ainsi qu'un dropout de 0,1. Enfin, pour obtenir un résultat au bon format, la tête du modèle a été remise sous la forme d'une couche à 2 nœuds avec pour fonction d'activation softmax.

Enfin, nous avons étudié l'impact du taux d'apprentissage et du nombre d'epoch.

2.1.3 Préparation des données et finetuning

Après avoir été téléchargé directement des bibliothèques de données tensorflow, les données ont été préparé en utilisant le tokenizer fournit par Huggingface, adapté à Bert. Nous avons utilisé le modèle pré-entrainé "bert-base-uncased" dont les caractéristiques sont disponibles sur Huggingface [Hugging Face, 2024] et l'optimiseur Adam. Après avoir apporté les modifications voulues aux couches suivant les hyperparamètres définis, le modèle est entrainé à nouveau sur l'ensemble d'entrainement grâce à la fonction fit de tensorflow. Certains modèle sont alors testés sur les ensembles de test grâce à la fonction evaluate de Tensorflow. Les résultats de perte et de prédiction sont alors enregistrés et analysés sur Excel.

2.2 GPT

Ensuite, nous avons décidé de réaliser une opération de finetuning similaire pour le modèle GPT d'OpenAI. Plus spécifiquement, nous avons sélectionné le modèle gpt-3.5-turbo qui est celui recommandé par OpenAI pour la plupart des tâches de finetuning. Cette section détaille méthodiquement les étapes suivies, depuis la préparation initiale des données jusqu'à l'évaluation finale du modèle sur un ensemble de données distinct.

2.2.1 Préparation et Transformation des Données

Le premier objectif était de transformer les données textuelles brutes des critiques de films IMDb en un format compatible avec l'entraînement d'un modèle de conversation basé sur

l'API d'OpenAI. Il était essentiel que chaque entrée soit structurée de manière à simuler une interaction utilisateur-système réelle, où le système est sollicité pour analyser le sentiment.

Le processus a débuté par l'extraction des critiques et de leurs sentiments associés, classés simplement en positif ou négatif. Chaque critique a été nettoyée pour éliminer les balises HTML et les caractères spéciaux, et normalisée pour uniformiser le format du texte. Suite à cela, chaque critique a été reformatée en une structure de conversation, où un 'utilisateur' présente la critique et le 'système' (le modèle) est incité à prédire le sentiment. On introduit également le rôle d'assistant pour forcer la réponse du modèle à être "positive" ou "negative". Pour cela l'assistant va introduire la réponse de la façon suivante: "The sentiment of the review is: ", et le système sera chargé de compléter la réponse.
Cette structuration est cruciale car elle aligne les données d'entraînement sur le format d'entrée attendu par le modèle lors de son utilisation réelle, optimisant ainsi la pertinence des apprentissages réalisés lors des phases subséquentes de finetuning.

2.2.2 Utilisation de l'API OpenAI pour le Finetuning

L'objectif ici était d'adapter le modèle préexistant GPT-3.5-turbo pour qu'il excelle spécifiquement dans la tâche de détermination de sentiment à partir de critiques textuelles, une capacité non nativement optimisée dans le modèle de base.
En réalité, le modèle pré-entraîné est de base parfaitement capable de comprendre une critique et de déterminer si elle est positive ou négative. Le but réel de l'entraînement est donc d'apprendre au modèle à donner une réponse dans le format attendu, en un seul mot "positive" ou "négative". Des données dans ce format peuvent permettre par la suite, par exemple à des compagnies, d'obtenir des données statistiques claires sur les critiques et feedback des utilisateurs.

Nous avons employé l'API fine-tuning d'OpenAI, configurant soigneusement les paramètres tels que le nombre d'époques, la taille des lots et le multiplicateur du taux d'apprentissage pour adapter le modèle aux spécificités de notre jeu de données.
Contrairement aux méthodes que nous avons employé pour le finetuning de BERT, l'API d'OpenAI n'autorise pas de contrôle précis sur le nombre de couches qui seront réentraînés (donc pas de couche gelée). Pour GPT, toutes les couches du modèle sont donc sujettes à l'apprentissage. Ce processus était accompagné d'une surveillance rigoureuse des performances du modèle et d'ajustements itératifs pour finement calibrer les paramètres en fonction des résultats observés.

3 Expériences

3.1 BERT

Les différentes expériences sur Bert ont été réalisé sur Google Colab Pro au moyen d'un GPU V100. En dehors de l'importation, la préparation des données a pris environ 10 minutes. Le finetuning a pris entre 30 et 40 minutes suivant les hyperparamètres. Les hyperparamètres non-étudiés ont été fixés à :

- taille d'échantillon : 6
- Longueur maximal d'entrée : 512 (le maximum pour Bert)

Tout les résultats, ainsi que les hyperparamètres correspondant peuvent être trouvés en annexe.

3.1.1 Nombre de couches gelées

Le nombre de couches gelés varie entre 0 (aucune) et 12 (toutes, c'est à dire le modèle sans finetuning). Moins il y a de couche gelés, plus le finetuning a d'impact sur le modèle.

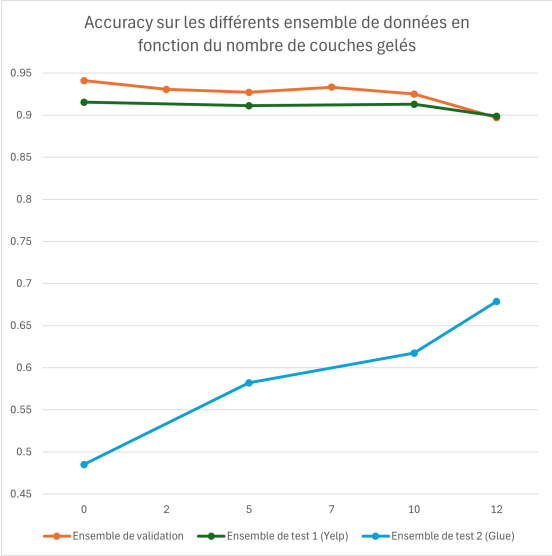


Figure 2: Précision en fonction du nombre de couches gelées

Comme on peut le voir sur la figure 2, assez logiquement, il y a une assez grande différence entre 10 et 12 couches gelées. Cela correspond à la différence entre aucun finetuning et un finetuning léger. Comme on peut le voir, le modèle est déjà très efficace en ce qui concerne la tâche demandée, mais le finetuning permet tout de même de gagner 4% de précision. De même, comme on peut s'y attendre, moins il y a de couches gelées, plus le modèle donne de bons résultats sur l'ensemble de validation.
Cependant, en termes de transférabilité, la précision sur l'ensemble de test 1 reste assez constante. Après le gain

initial entre avec et sans finetuning, la précision se stabilise sans gain visible vers 0,91. Le fait que la précision sur l'ensemble de validation continue d'augmenter illustre un probable surentraînement du modèle en dessous de 10 couches gelées.
En ce qui concerne le deuxième ensemble de test, Bert sans finetuning atteint 67% de précision. Le pré-entraînement de Bert est donc initialement moins bien adapté à cette tache-ci mais l'exécute tout de même suffisant bien pour pouvoir observer le scénario d'oubli catastrophique. Moins il y a de couche gelé, pire sont les résultats. On pouvait s'y attendre, mais à 0 couche gelée, le modèle tombe à 50% de précision. Cette valeur correspond à l'espérance statistique d'un modèle complètement aléatoire et signifie donc que le modèle n'est plus apte à effectuer cette tâche. C'est un oubli catastrophique complet. Les performances descendent de manière assez régulière. Puisque l'on a vu précédemment que les gains deviennent marginaux en dessous de 10 couches gelés dû au surentraînement, 10 ou 11 couches gelés paraissent être le meilleur choix pour équilibrer la performance et l'oubli.

3.1.2 Nombre de couches additionnelles

De nouveaux modèles possédant une couche supplémentaire ont été créé et testés avec un différend nombre de couches gelés. Au vu des conclusions précédentes, nous nous sommes particulièrement concentrés sur les modèles à 10 et 12 couches gelées. Nous nous sommes arrêté à deux couches dû à l'absence de différences significatives entre les résultats avec une ou deux couches supplémentaires.

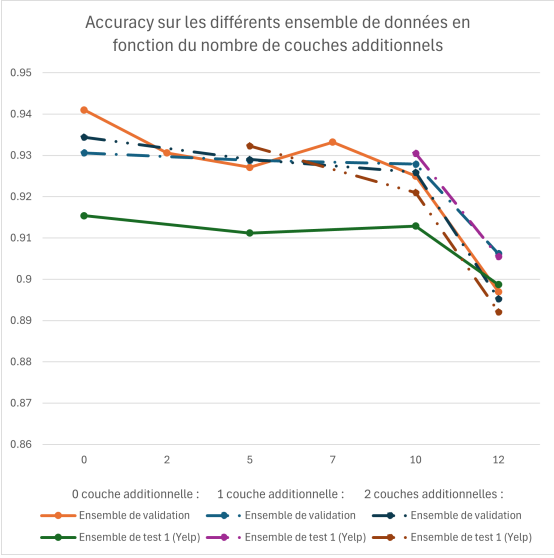


Figure 3: Précision en fonction du nombre de couches additionnelles

Comme on peut le voir sur la figure 3, le nombre de couches supplémentaires n'a pratiquement aucun impact sur l'ensemble de validation. Par contre, l'ajout d'une couche supplémentaire augmente fortement la transférabilité du modèle. La seconde couche n'a, par contre, aucun effet visible.
L'ajout d'une couche supplémentaire réduit de presque 15% la précision sur l'ensemble de test 2 quand toute les couches sont gelées (c'est à dire que seule la couche supplémentaire est entraînée). En revanche, cette perte se superpose partiellement à celle dû aux couches non-gelés. Pour 1 couche additionnelle, les précision entre 10 et 12 couches gelées sont assez proche et seulement légèrement inférieur à la précision en l'absence de couches additionnelles. L'idéal serait donc de lier les deux méthodes pour un rapport efficacité/oubli catastrophique optimal. Dans l'exemple de cette étude, les meilleurs hyperparamètres seraient : 10 couches gelés et 1 couche additionnelle.

3.1.3 Taux d'apprentissage

Nous avons testé deux taux d'apprentissages: 2e-5 et 3e-5. Toutes les expériences précédentes ont été réalisés avec un taux d'apprentissage de 2e-5. Pratiquement aucune différence notable n'est visible entre les deux taux d'apprentissages sans couches supplémentaires et avec au moins 5 couches gelées. En revanche, le taux d'apprentissage de 3e-5 montre une très forte baisse des résultats sur tous les ensembles (sauf celui d'entraînement) quand moins de couches sont gelées et/ou que des couches supplémentaires sont ajoutés. Cela est probablement dû au surentraînement. Le taux d'apprentissage de 2e-5 sera donc retenu dans cet exemple.

3.1.4 Nombre d'epoch

Exécuter plus d'une epoch est extrêmement long. Pour des raisons de temps disponible, nous ne sommes donc pas allés au-delà de deux epochs. Et nous l'avons testé en ne faisant varier que les couches gelées (sans couches additionnelles et pour un taux d'apprentissage de 2e-5).
Les modèles entraînés sur une ou deux epochs ont donné des résultats quasiment identiques. En raison des ressources supplémentaires nécessaire à une seconde epoch, le choix se portera donc sur une seule epoch.

3.2 GPT

Les expériences réalisées sur le modèle GPT ont été plus limitées que celles sur BERT pour les raisons suivantes:

1. **Manque de contrôle sur le finetuning:** Bien que nous pouvons gérer les hyperparamètres du job de finetuning tels que le learning rate, la taille de batch et le nombre d'epoch, il n'est pas possible de spécifiquement décider quelles couches seront ré-entraînés: elles le sont toutes. Nous n'aurons pas donc de comparaison pour le gel de couches avec BERT, nous comparerons donc les performances avec 0 couches gelées. Pour les mêmes

raisons, il n'est pas possible d'ajouter des couches supplémentaires au modèle.

2. **Coût d'utilisation de l'API:** Si nous avions décidé d'utiliser un modèle GPT gratuit, nous n'aurions pu expérimenter qu'avec le modèle "gpt-2". Cependant, cela n'aurait pas été pertinent car c'est un modèle très dépassé et bien moins performant par rapport aux modèles plus récents (à partir de gpt-3). Pour cette raison, nous avons fait le choix d'employer le modèle gpt-3.5-turbo, ce qui impliquait l'utilisation de l'API pour le finetuning et les inférences.
   Chaque requête à l'API est coûteuse, et le prix ne se compte pas en nombre de données mais en nombre de tokens. Les prompts de nos datasets étant très longs (parfois plusieurs paragraphes), il a fallu considérablement réduire la taille des datasets utilisés ainsi que le nombre d'expériences réalisées.
   Pour cette raison, nous n'avons pas poursuivi les tests d'hyperparamètres jusqu'au bout de l'entraînement si les résultats ne semblaient pas concluants sur les premières étapes. Nous avons fait le choix d'interrompre ces entraînements à mi-chemin pour trouver de façon empirique des hyperparamètres optimaux.

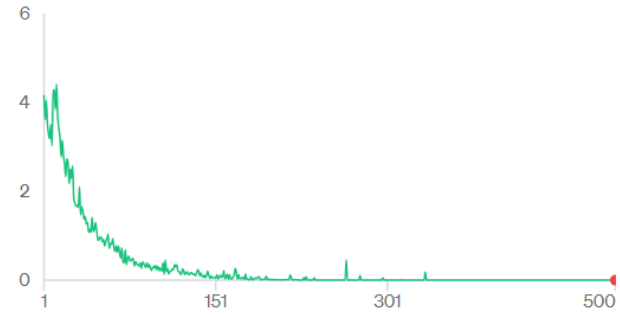
3.2.1 Entraînement

L'entraînement a été réalisé sur le dataset IMDB_movie_reviews, qui contient 50,000 critiques de films libellées "positive" ou "négative". Comme précisé précédemment, il a fallu réduire la taille de ce dataset pour le finetuning. Nous avons donc aléatoirement sélectionné 5,000 exemples que nous avons découpé en un set d'entraînement et un set de validation avec les proportions 80%/20%.
Grâce à l'API nous avons pu importer ces fichiers sur les serveurs d'OpenAI afin de démarrer le job de finetuning sur leur modèle "gpt-3.5-turbo-0125".
De façon empirique, en observant les premières étapes de l'entraînement, nous avons sélectionné les paramètres d'entraînement suivants :

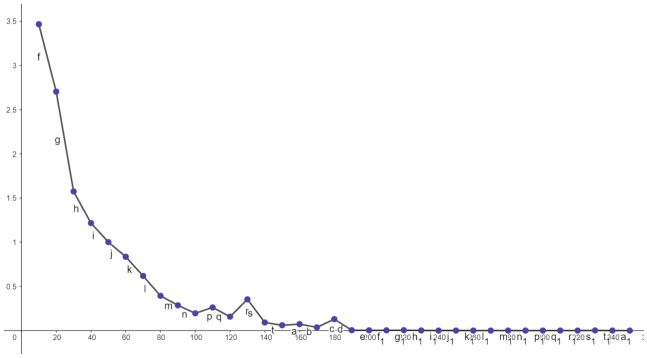
- Epochs: 1
- Batch_size: 8
- Learning_rate_multiplier: 0.1

3.2.2 Résultat

Voici les résultats obtenus lors de l'entraînement sur une epoch:



(a) Training loss



(b) Validation loss

Figure 4: Perte d'entraînement et de validation en fonction du nombre d'étapes

On remarque que la perte d'entraînement tombe assez rapidement à 0, ce qui pourrait indiquer un phénomène de surentraînement. Cependant, la perte de validation tombe également à 0, c'est donc un bon signe que le modèle n'est pas surentraîné.

Nous avons ensuite évalué le modèle sur le dataset "yelp_polarity". Cela permet de voir si le modèle est capable de se généraliser à d'autres type de critiques que les critiques de films.
L'utilisation de l'API étant tout aussi coûteuse pour les inférences que pour l'entraînement, il a également fallu réduire le nombre de données utilisées pour l'évaluation. Nous avons donc stoppé l'évaluation à 4500 exemples.
Nous avons relevé la précision du modèle ainsi que le pourcentage de réponses au format incorrect, c'est à dire les réponses qui ne sont ni "positive" ni "négative". Nous avons réalisé la même évaluation sur le modèle non finetuné pour comparaison.

Modèle	Précision	format incorrect
Modèle finetuné	93%	5.8%
gpt-3.5-turbo	73%	26.3%

Table 1: Résultats sur Yelp_Polarity

3.2.3 Analyse

Les données présentées dans le tableau Table 1 montrent qu'avec 93% de précision et 5.8% de réponses au format incorrect, le modèle se trompe entre "positive" et "négative" dans 1.2% des cas.
Le modèle non finetuné, quant à lui, ne se trompe entre "positive" et "négative" que dans 0.7% des cas mais présente 26.3% de réponses au format incorrect.
Pour le modèle finetuné, les réponses incorrectes sont généralement "neutral" ou "mixed" ce qui montre que le modèle n'a pas totalement appris à donner des réponses binaires. Ces résultats suggèrent qu'utiliser des datasets qui présentent 3 possibilités de vérités-terrains ("positive", "neutral" ou "négative") donnerait sûrement de meilleurs résultats.
L'autre réponse incorrect la plus courante est "the". Cela suggère que, bien que nous forcions les réponses à être d'une longueur de 1 Token, le modèle tente de répondre par une phrase (on peut conjecturer que la réponse serait du type "the sentiment is ..."). Cela ne semble pas se produire durant l'entraînement, même sur le set de validation, et pourrait signifier qu'il y a une mauvaise gestion du rôle de l'assistant lors des inférences.
Les autres réponses incorrectes sont moins facilement explicables: nous observons "fr", "pr", "nost" et "dis", chacun à plusieurs occurrences, qui ne sont ni des débuts des phrases ni même des mots corrects. Cela ne se produit pas avec le modèle GPT non finetuné, qui présente des réponses incorrectes du type "neutral", "mixed", "overall", "it", "I" ou "the", qui semblent être des débuts de phrases et être cohérents avec le fait que le modèle n'est pas entraîné à donner une réponse en un mot. Ce phénomène, couplé au fait qu'on passe de 0.7% à 1.2% d'erreurs entre "positive" et "négative", pourrait indiquer un début d'oubli catastrophique, ce qui s'expliquerait par le fait que nous réentraînons toutes les couches du modèle. il serait intéressant d'observer si un gel des couches pourrait adresser ces erreurs, mais ce n'est actuellement pas une possibilité avec le modèle étudié et l'API utilisée.

3.3 Comparaison des modèles

Pour BERT, le modèle ayant montré les meilleures performances sur yelp_polarity avec une précision de 93.23% est celui utilisant les hyperparamètres suivants:

- Epochs: 1
- Learning_rate: 0.00002
- Couches gelées: 5
- Couches additionelles: 2

Il est intéressant de noter que bien que ce modèle atteigne des performances comparables à notre GPT finetuné sur le dataset yelp_polarity, il avait en fin d'entraînement une loss de 0.24 sur l'ensemble d'entraînement est de 0.19 sur l'ensemble de validation de IMDB_movie_reviews, contre 0 pour les deux ensembles pour GPT. C'est une comparaison difficile car la taille des ensembles d'entraînement et de validation utilisés pour GPT étaient réduites d'un facteur 10, mais cela pourrait indiquer une capacité accrue du modèle BERT a se généraliser à d'autres types de critiques au lieu de parfaitement apprendre à classifier des critiques de films.

Une autre observation qui peut être faite est que lorsqu'aucune couche n'est gelée ou ajoutée pour BERT (donc l'ensemble des couches sont sujettes au finetuning, comme pour GPT), la précision atteinte sur yelp_polarity n'est que de 91.5%.
Cela indique deux choses: premièrement, le gèle et l'ajout de couche est effectivement une méthode efficace qui permet d'améliorer les performances d'un modèle lors d'un finetuning. Deuxièmement, nous pouvons conjecturer qu'avec ces techniques, si elles étaient applicables au modèle GPT, permettrait d'obtenir des performances supérieures à celles observées.

4 Critique de l'approche et conclusion

Dû au manque de temps par rapport à la durée d'exécution d'un entrainement, il n'a pas été possible d'étudier l'impact de la taille d'échantillon. Le nombre de tests sur les paramètres étudiés a également été très limité. Un ou deux entrainements de modèle seulement ont pu être exécuté pour chaque ensemble d'hyperparamètres. Cela permet de dégager des tendances, mais est insuffisant pour une étude statistique rigoureuse. Les résultats sur lesquels se basent nos conclusions sont ainsi soumis à une forte variabilité aléatoire pouvant grandement altérer l'analyse.
Le modèle de Bert utilisé est également assez simpliste. L'étude d'un modèle plus avancé et possédant plus de paramètres pourrait révéler d'autres impacts ou affiner nos conclusions.

Une autre des critiques principales que nous pouvons faire vis à vis de notre projet est la difficulté que nous avons eu à produire une comparaison juste entre les modèles BERT et GPT. En effet, pour des raisons purement financières en raison du coût d'utilisation de l'API d'OpenAI, nous avons dû considérablement réduire le nombre d'expériences réalisées ainsi que la taille et le nombre de données utilisées dans les entraînements.
Ainsi, les modèles basés sur BERT ont pu être entraînés sur un ensemble d'entraînement/validation de 50,000 critiques de films contre seulement 5,000 pour GPT. De même pour les évaluations sur yelp_polarity, qui ont dû être raccourcie pour GPT.
Nous avons pu tester une large gamme d'hyperparamètres pour BERT, mais nous avons dû fonctionner de manière empirique pour GPT en observant le comportement du modèle sur les premières centaines d'étapes d'entraînement.

Cela a eu pour conséquence de ne pas permettre de fournir des conditions expérimentales égales pour les deux modèles afin de faire une juste comparaison.
Pour cette raison, bien que nos résultats semblent pencher en faveur du finetuning de GPT pour les performances sur l'analyse de sentiment, il semble plus pertinent d'utiliser BERT pour cette tâche. C'est une méthode qui est non seulement gratuite (bien que l'entraînement soit limité par le Hardware de l'utilisateur ou l'utilisation de services comme Google Collab) mais qui permet d'avoir un meilleur contrôle sur le finetuning et bien plus d'options de customisation avec le gel ou l'ajout de couches au modèle.

En conclusion, nous avons pu réaliser un grand nombre d'expérimentations pour étudier les techniques de finetuning de modèles Transformers pour la tâche de l'analyse de sentiments. Voici les observations que nous avons pu tirer de nos travaux:

1. Les méthodes de gèle et d'ajout de couche lors du finetuning semblent améliorer les performances du modèle BERT sur l'analyse de Sentiments. Notre cas touve son efficacité optimal pour les hyperparamètres suivants :
    - Epochs : 1
    - Learning_rate : 0.00002
    - Couches gelées : 10
    - Couche additionelle : 1
2. BERT et GPT, bien qu'ils présentent des architectures Transformers différentes (respectivement Encoder-Only et Decoder-only), permettent tout deux d'obtenir des résultats très satisfaisants quant à l'analyse de Sentiments sur des critiques d'utilisateurs.
3. Les observations faites concernant les réponses au format incorrect des modèles (de type "mixed" ou "neutral") indiquent qu'il serait judicieux de tester cette méthode sur des Datasets incluant 3 labels pour les vérités terrains: "positve", "negative" ou "neutral".
4. Bien que GPT présente des performances satisfaisante et encourageante, les limitations induites par l'utilisation de l'API d'OpenAI, le coût d'utilisation de celle-ci et le manque de contrôle sur le finetuning ne font pas de GPT le choix de modèle idéal pour cette tâche, s'il s'agit d'un projet de recherche.
   GPT pourrait cependant convenir à un contexte d'utilisation réel (par exemple d'étude de feedback de produit) où le modèle serait finetuné d'une seule manière, sur le Dataset entier pour ensuite être déployé à grande échelle.

References

[Hugging Face, 2024] Hugging Face. google-bert/bert-base-uncased. https://huggingface.co/google-bert/bert-base-uncased, 2024. Accessed: 2024-04-28.

[Krugmann and Hartmann, 2024] J. O. Krugmann and J. Hartmann. Sentiment analysis in the age of generative ai. *Customer Needs and Solutions*, 11(1), 2024. 10.1007/s40547-024-00143-4.

[Luo *et al.*, 2023] Y. Luo, Z. Yang, F. Meng, Y. Li, J. Zhou, and Y. Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.

[Moonat, 2021] D. Moonat. Fine-tune bert model for sentiment analysis in google colab. https://www.analyticsvidhya.com/blog/2021/12/fine-tune-bert-model-for-sentiment-analysis-in-google-colab/?fbclid=IwAR0YclRjpJy-Ezg0657aez5wpp-DBpgVK15h1ymFHoj_vi7IFlKKTWbd7zA, 2021. Accessed: 2024-04-28.

[Nguyen *et al.*, 2020] Q. T. Nguyen, T. L. Nguyen, N. H. Luong, and Q. H. Ngo. Fine-tuning bert for sentiment analysis of vietnamese reviews. In *2020 7th NAFOSTED Conference on Information and Computer Science (NICS)*. IEEE, 2020.

[Pang *et al.*, 2002] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Sentiment classification using machine learning techniques. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2002)*, 2002.

[Raschka, 2023] S. Raschka. Finetuning llms efficiently with adapters. https://magazine.sebastianraschka.com/p/finetuning-llms-with-adapters, 2023. Accessed: 2024-05-20.

[Takyar, 2023] A. Takyar. Fine-tuning pre-trained models for generative ai applications. https://www.leewayhertz.com/fine-tuning-pre-trained-models/, 2023. Accessed: 2024-02-28.

[TensorFlowDatasets, 2024] TensorFlowDatasets. Ensembles de données. https://www.tensorflow.org/datasets/catalog/overview?hl=fr, 2024. Accessed: 2024-04-28.

A Annexe - Résultats supplémentaires

Table 2: Résultats des différentes expériences de finetuning sur Bert

Expérience numéro	0	1	2	3	4	5
		Hyperparamètres				
Nombre d'epoch	Sans finetuning	1	1	1	1	1
Taux d'apprentissage	-	0.00002	0.00002	0.00002	0.00002	0.00002
Nombre de couches gelées	-	0	2	5	7	10
Nombre de couches additionelles	-	0	0	0	0	0
		Résultats				
		Ensemble d'entrainement				
Loss	0.3607	0.2437	0.2351	0.2386	0.2386	0.2558
Accuracy	0.8459	0.9025	0.9050	0.9057	0.9053	0.8945
		Ensemble de validation				
Loss	0.2510	0.1669	0.1957	0.1914	0.1693	0.1907
Accuracy	0.8970	0.9410	0.9306	0.9271	0.9332	0.9250
		Ensemble de test 1 (Yelp)				
Loss	0.2590	0.2333	-	0.2173	-	0.2054
Accuracy	0.8987	0.9154	-	0.9112	-	0.9129
		Ensemble de test 2 (Glue)				
Loss	0.6148	0.8786	-	0.7777	-	0.8248
Accuracy	0.6788	0.4851	-	0.5820	-	0.6174

Table 3: Résultats des différentes expériences de finetuning sur Bert (suite)

Expérience numéro	6	7	8	9
Hyperparamètres				
Nombre d'epoch	1	1	1	1
Taux d'apprentissage	0.00002	0.00002	0.00002	0.00002
Nombre de couches gelées	0	5	10	12
Nombre de couches additionelles	1	1	1	1
Résultats				
Ensemble d'entrainement				
Loss	0.2541	0.2495	0.2666	0.2084
Accuracy	0.8996	0.9025	0.8883	0.9192
Ensemble de validation				
Loss	0.1867	0.1827	0.1868	0.2351
Accuracy	0.9306	0.9288	0.9279	0.9062
Ensemble de test 1 (Yelp)				
Loss	-	-	0.1901	0.2443
Accuracy	-	-	0.9305	0.9055
Ensemble de test 2 (Glue)				
Loss	-	-	0.8002	0.7190
Accuracy	-	-	0.5724	0.5665

Table 4: Résultats des différentes expériences de finetuning sur Bert (suite)

Expérience numéro	10	11	12	13
Hyperparamètres				
Nombre d'epoch	1	1	1	1
Taux d'apprentissage	0.00002	0.00002	0.00002	0.00002
Nombre de couches gelées	0	5	10	12
Nombre de couches additionelles	2	2	2	2
Résultats				
Ensemble d'entrainement				
Loss	0.2653	0.2429	0.2598	0.3654
Accuracy	0.8836	0.8985	0.8903	0.8342
Ensemble de validation				
Loss	0.1762	0.1884	0.2012	0.2553
Accuracy	0.9344	0.9290	0.9259	0.8953
Ensemble de test 1 (Yelp)				
Loss	-	0.1791	0.2152	0.2660
Accuracy	-	0.9323	0.9210	0.8921

Table 5: Résultats des différentes expériences de finetuning sur Bert (suite)

Expérience numéro	14	15	16
Hyperparamètres			
Nombre d'epoch	1	1	1
Taux d'apprentissage	0.00003	0.00003	0.00003
Nombre de couches gelées	0	5	10
Nombre de couches additionelles	0	0	0
Résultats			
Ensemble d'entrainement			
Loss	0.2536	0.2414	0.2479
Accuracy	0.8986	0.9038	0.8998
Ensemble de validation			
Loss	0.1717	0.1718	0.1863
Accuracy	0.9358	0.9314	0.9282
Ensemble de test 1 (Yelp)			
Loss	-	0.2173	0.2185
Accuracy	-	0.9112	0.9095
Ensemble de test 2 (Glue)			
Loss	-	0.7777	0.7182
Accuracy	-	0.5820	0.6453

Table 6: Résultats des différentes expériences de finetuning sur Bert (suite)

Expérience numéro	17	18	19
Hyperparamètres			
Nombre d'epoch	1	1	1
Taux d'apprentissage	0.00003	0.00003	0.00003
Nombre de couches gelées	0	5	10
Nombre de couches additionelles	1	1	1
Résultats			
Ensemble d'entrainement			
Loss	0.2740	0.2487	0.2568
Accuracy	0.8912	0.9015	0.8986
Ensemble de validation			
Loss	0.2444	0.1771	0.1812
Accuracy	0.9022	0.9310	0.9300
Ensemble de test 1 (Yelp)			
Loss	-	0.2261	0.1875
Accuracy	-	0.9131	0.9276

Table 7: Résultats des différentes expériences de finetuning sur Bert (suite et fin)

Expérience numéro	20	21	22
Hyperparamètres			
Nombre d'epoch	2	2	2
Taux d'apprentissage	0.00002	0.00002	0.00002
Nombre de couches gelées	0	5	10
Nombre de couches additionelles	0	0	0
Résultats			
Ensemble d'entrainement			
Loss	0.1339	0.1258	0.1513
Accuracy	0.9526	0.9561	0.9438
Ensemble de validation			
Loss	0.1909	0.2229	0.1853
Accuracy	0.9336	0.9294	0.9332