26/02/2024

Université : Lumières Lyon2

Projet : fine-tuning du modèle Camembert pour l'analyse de sentiments.

Réalisé par ZIDAN Lama & ZIDAN Loubna

Enseignantes :

POURKOSINA Irina & CHRETIEN Stéphane

Table des matières

[1.Introduction aux Transformers et à l'Auto-Attention 2](#_Toc159879502)

[1.1.Que ce que c'est les Transformers et l’auto-attention ? 2](#_Toc159879503)

[1.2.limitations des RNN et LSTM et Révolution apportée par l'Auto-Attention 2](#_Toc159879504)

[1.3.Fonctionnalité Encodeur-Décodeur : 3](#_Toc159879505)

[2. Présentation de Camembert : 5](#_Toc159879506)

[2.1.Pourquoi Camembert est Adapté à la Compréhension du Texte ? Encodeur-Encodeur : 5](#_Toc159879507)

[2.2.Architecture Camembert-Base 5](#_Toc159879508)

[2.3.Tâche Initiale et Dataset d'Entraînement 6](#_Toc159879509)

[3.Cadre expérimental : 7](#_Toc159879510)

[3.1. Nouvelle tâche d'analyse de sentiments 7](#_Toc159879511)

[3.2. Dataset et prétraitement 7](#_Toc159879512)

[3.3.Sélection et Justification des Hyperparamètres 8](#_Toc159879513)

[4. Résultats et discussion 10](#_Toc159879514)

[4.1.Résultats d’entrainement : 10](#_Toc159879515)

[4.2.Analyse des Courbes de Perte d'Entraînement et de Validation : 11](#_Toc159879516)

[4.3.Évaluation des Performances du Modèle 12](#_Toc159879517)

[5.Conclusion : 14](#_Toc159879518)

## 1.Introduction aux Transformers et à l'Auto-Attention

### 1.1.Que ce que c'est les Transformers et l’auto-attention ?

Les Transformers sont des modèles de pointe en traitement automatique du langage naturel (NLP), introduits par Vaswani et al. en 2017. Ils ont apporté une révolution dans la compréhension et la génération du langage par les machines grâce à leur architecture basée sur un mécanisme novateur : l'auto-attention. Cette innovation permet aux Transformers de surpasser les modèles antérieurs, tels que les réseaux de neurones récurrents (RNN) et les Long Short-Term Memory (LSTM), en termes d'efficacité et de compréhension des relations complexes au sein des textes.

L'auto-attention est au cœur des Transformers. Ce mécanisme permet au modèle de peser l'importance de chaque mot dans un contexte donné, en fonction de tous les autres mots de la séquence. Cela signifie que, contrairement aux méthodes antérieures qui traitent les mots séquentiellement, les Transformers peuvent évaluer l'ensemble de la séquence en une fois, permettant une compréhension plus riche et contextuelle du texte.

Avant l'introduction du mécanisme d'auto-attention dans les architectures Transformer, les modèles de traitement du langage naturel (NLP) reposaient principalement sur les réseaux de neurones récurrents (RNN) et leurs variantes, comme les LSTM (Long Short-Term Memory) et les GRU (Gated Recurrent Units).

Ces modèles traitent le texte séquentiellement, c'est-à-dire mot par mot, en maintenant un "état caché" qui est censé capturer les informations pertinentes de tous les mots précédents à chaque étape.

### 1.2.limitations des RNN et LSTM et Révolution apportée par l'Auto-Attention

Les RNN standards calculent l'état caché d'un mot en fonction de l'état précédent et du mot actuel, utilisant des paramètres du modèle et une fonction d'activation non linéaire. Les LSTM améliorent cette approche en intégrant des portes de contrôle pour gérer les informations et les dépendances à long terme plus efficacement.

Cependant, les RNN et LSTM présentent des limitations, notamment la difficulté de traiter de longues séquences due au problème de disparition ou d'explosion des gradients. De plus, leur nature séquentielle limite la parallélisation des calculs, ce qui réduit la vitesse d'entraînement et d'inférence sur des ordinateurs modernes.

L'auto-attention, introduite avec les Transformers, a révolutionné le NLP en permettant à chaque mot dans une séquence d'être traité en fonction de tous les autres mots de la séquence simultanément. Mathématiquement, pour une séquence d'entrée, l'auto-attention calcule un ensemble de scores d'attention qui déterminent à quel point chaque mot doit "accorder d'attention" à tous les autres mots. Ces scores sont ensuite utilisés pour pondérer une combinaison linéaire des représentations des mots, produisant une nouvelle représentation qui capte les contextes pertinents pour chaque mot.

Cette capacité à traiter toutes les parties d'une séquence en parallèle et à capturer des dépendances complexes et à longue distance sans être limitée par la structure séquentielle des RNN a permis des améliorations substantielles dans presque toutes les tâches de NLP. De plus, l'architecture Transformer facilite l'entraînement de modèles plus grands sur des séquences plus longues, grâce à une efficacité computationnelle accrue, ouvrant la voie à des modèles pré-entraînés de grande envergure comme BERT, GPT, et leurs successeurs, qui dominent aujourd'hui le champ du NLP.

Une image contenant texte, capture d’écran, nombre, ligne

Description générée automatiquement

Pour mieux comprendre le contexte d'un mot dans une phrase. Prenons la phrase en exemple : "L'animal n'a pas traversé la rue parce qu'il était trop fatigué". La question est de savoir à quoi "il" se réfère dans la phrase : est-ce à la rue ou à l'animal ? Pour un humain, la réponse est évidente, il s'agit de l'animal. Cependant, pour un algorithme, ce n'est pas si simple.

L'auto-attention permet au modèle de traiter le mot "il" en associant son sens au mot "animal" plutôt qu'à "rue". En d'autres termes, alors que le modèle lit chaque mot de la phrase, il peut "regarder" en arrière vers d'autres mots pour collecter des indices qui peuvent mener à une meilleure compréhension et codage du mot actuellement traité.

## 1.3.Fonctionnalité Encodeur-Décodeur :

La structure encodeur-décodeur est une caractéristique clé des Transformers, facilitant une large gamme de tâches de NLP. L'encodeur traite le texte d'entrée, capturant les informations contextuelles, tandis que le décodeur génère une sortie séquentielle. Cette conception permet aux Transformers d'être utilisés aussi bien pour la compréhension que pour la génération de texte, rendant les modèles extrêmement polyvalents.

**Une image contenant capture d’écran, ligne, diagramme, Parallèle

Description générée automatiquement**

Dans cet exemple, le modèle traduit la phrase française "Je suis étudiant" en anglais "I am a student". Voici comment cela fonctionne :

* Encodeur : La phrase d'entrée est d'abord convertie en vecteurs par le processus d'embedding. Ces vecteurs passent ensuite séquentiellement à travers plusieurs couches d'encodeurs, chacune effectuant des opérations complexes, comme l'auto-attention et la normalisation, pour transformer ces vecteurs en un nouveau jeu de vecteurs qui capturent le contexte de chaque mot dans la phrase. Même si la structure de chaque couche d'encodeur est identique, elles n'utilisent pas les mêmes poids (ce qui signifie qu'elles apprennent des aspects différents des données).
* Décodeur : Chaque couche du décodeur reçoit le jeu de vecteurs de l'encodeur et travaille avec les vecteurs générés par la couche précédente du décodeur. Les décodeurs utilisent également l'auto-attention, mais ils ont une attention supplémentaire qui leur permet de se concentrer sur des parties spécifiques des vecteurs d'entrée de l'encodeur. Ces opérations permettent au décodeur de générer la traduction un mot à la fois, en tenant compte de ce qui a déjà été généré et de l'entrée complète de l'encodeur.
* Sortie : La sortie de la dernière couche de décodeur est alors transformée en mots dans la langue cible, ici, l'anglais.

Les flèches allant de chaque couche d'encodeur à chaque couche de décodeur, illustrant que l'information de l'encodeur est utilisée à chaque étape du processus de décodage pour aider à générer la traduction la plus précise.

## 2. Présentation de Camembert :

### 2.1.Pourquoi Camembert est Adapté à la Compréhension du Texte ? Encodeur-Encodeur :

La structure à encodeur seul de Camembert lui permet de se concentrer sur l'extraction d'une représentation riche et nuancée du texte d'entrée. Chaque couche d'encodeur construit une compréhension de plus en plus riche du texte, en tenant compte du contexte de chaque mot par rapport à tous les autres mots du texte. Cela permet à Camembert de capter des subtilités sémantiques et syntaxiques spécifiques à la langue française, grâce aussi à son entraînement sur un vaste corpus de texte en français.

En effet il existe une large application de Camembert dans le NLP :

Les capacités de Camembert le rendent idéal pour un large éventail d'applications en NLP, notamment : Classification de texte, Analyse de sentiments, Reconnaissance d'entités nommées, Compréhension de texte

En résumé, Camembert se distingue par sa capacité à comprendre et analyser le texte sans se focaliser sur la génération de texte. Cette spécialisation en fait un outil puissant pour de nombreuses applications en NLP, spécifiquement adaptées au français.

Camembert est un modèle de traitement automatique du langage naturel (NLP) qui s'inspire de l'architecture Transformer. Sa particularité réside dans sa structure composée uniquement d'encodeurs, à la différence des architectures encodeur-décodeur utilisées dans d'autres modèles tels que BERT ou GPT. Cette spécificité confère à Camembert des capacités adaptées à l'analyse et à la compréhension du texte, notamment pour le français.

En effet, en se basant sur une architecture à encodeur seul, le modèle se concentre sur l'analyse du texte d'entrée pour en extraire des caractéristiques et des informations pertinentes. Sans la partie décodeur, le modèle n'est pas conçu pour générer du texte, mais plutôt pour comprendre et interpréter le texte d'entrée. Cela le rend particulièrement efficace pour des tâches comme la classification de texte (par exemple, déterminer si un commentaire est positif ou négatif), l'analyse de sentiments, la reconnaissance d'entités nommées (identifier les noms de personnes, d'organisations, de lieux dans un texte), ou encore la réponse à des questions (question answering) basées sur un corpus donné.

### 2.2.Architecture Camembert-Base

Camembert-Base est un modèle de traitement du langage naturel spécifiquement conçu pour comprendre la langue française, s'inspirant de l'architecture BERT-base. Il est structuré autour de 12 couches d'encodeurs, chacune équipée de 768 unités cachées et de 12 têtes d'attention. Cette architecture riche et complexe lui permet d'analyser le texte en profondeur, en capturant les subtilités et les nuances de la langue française à différents niveaux d'abstraction.

Embeddings Camembert :

Les embeddings de Camembert sont des vecteurs denses qui servent à représenter les mots dans un espace continu où des mots de sens similaire sont positionnés de manière proche les uns des autres. En plus des mots eux-mêmes, Camembert utilise également des embeddings pour tenir compte de la position des mots dans les phrases, ce qui est essentiel étant donné que la signification des mots peut changer en fonction de leur emplacement dans une séquence de texte. Ces embeddings positionnels permettent au modèle de saisir l'ordre et la structure syntaxique, qui sont cruciaux pour la compréhension grammaticale.

Camembert utilise également des embeddings spécifiques pour distinguer différents types de tokens dans une phrase. Par exemple, dans le cas où une phrase contient des mots clés spéciaux ou des séparateurs de phrases, le modèle peut utiliser ces embeddings de type de token pour comprendre et traiter différemment ces éléments par rapport aux mots ordinaires.

* Encodeur Camembert : L'encodeur est le cœur du modèle, avec 12 couches successives qui traitent le texte d'entrée. Chaque couche utilise l'auto-attention pour permettre au modèle de considérer l'ensemble de la séquence d'entrée et de comprendre le contexte de chaque mot par rapport aux autres.
* Attention Camembert : Chaque couche d'encodeur intègre un mécanisme d'attention qui évalue l'importance relative de chaque mot par rapport aux autres dans le texte. Cela permet au modèle de se concentrer sur les parties les plus pertinentes du texte pour une tâche donnée.
* Intermédiaire et Sortie : Après le mécanisme d'attention, chaque couche d'encodeur comporte une couche intermédiaire qui transforme les représentations, suivie d'une couche de sortie qui consolide les représentations pour la couche suivante. L'activation GELU est utilisée comme fonction d'activation intermédiaire, favorisant une modélisation non linéaire efficace.

### 2.3.Tâche Initiale et Dataset d'Entraînement

Le modèle Camembert, une variante spécifique au français des architectures de traitement automatique du langage naturel basées sur les Transformers, a été initialisé grâce à un processus de pré-entraînement sur le corpus OSCAR. OSCAR est une collection substantielle de textes en français obtenus à partir d'internet, représentant une diversité de contextes et de styles linguistiques.

Durant la phase de pré-entraînement, le modèle Camembert a été soumis à une tâche d'apprentissage auto-supervisée : la prédiction de tokens masqués. Cette tâche consiste à occulter aléatoirement certains mots d'un texte et à entraîner le modèle à les prédire en se basant sur leur contexte. Cela permet au modèle de développer une compréhension profonde des relations contextuelles entre les mots dans la langue française.

Grâce à cette méthode, Camembert apprend à générer des représentations vectorielles, ou embeddings, qui capturent la signification des mots dans des contextes variés. Cet entraînement préliminaire est crucial, car il équipe le modèle de la capacité à saisir la nuance et la complexité de la langue française, le rendant ensuite apte à être affiné, ou "fine-tuned", pour des tâches spécifiques comme l'analyse de sentiments, la reconnaissance d'entités nommées, la classification de texte, et bien plus.

Le pré-entraînement sur un large corpus tel que OSCAR assure que Camembert dispose d'une base de connaissances linguistiques étendue et robuste, essentielle pour sa performance lors des tâches ultérieures de traitement du langage.

## 3.Cadre expérimental :

L'objectif principal de notre projet est de spécialiser le modèle Camembert-base, un outil de pointe en traitement automatique du langage naturel conçu spécifiquement pour le français, pour la tâche d'analyse de sentiments. Cette initiative cherche à habilité le modèle à classifier de manière efficace des textes inédits en français en sentiments positifs ou négatifs. L'importance de l'analyse des sentiments s'étend à diverses applications, allant de la gestion de la réputation en ligne à l'analyse des feedbacks clients, soulignant la valeur de notre travail. L'approche choisie, le fine-tuning, implique une calibration fine des paramètres du modèle pré-entraîné sur un corpus généraliste, optimisant ainsi sa performance pour notre tâche spécifique. Cette méthode permet de capitaliser sur la compréhension approfondie de la langue française par Camembert tout en l'adaptant aux nuances fines de l'expression sentimentale.

### 3.1. Nouvelle tâche d'analyse de sentiments

L'analyse de sentiments est une tâche cruciale dans le traitement automatique des langues, permettant de détecter et d'interpréter les nuances sentimentales exprimées dans les textes. Pour Camembert, un modèle déjà entraîné sur un large éventail de textes en français, le défi consiste à réajuster ses paramètres de manière à ce qu'il puisse reconnaître ces nuances sentimentales avec une grande précision. Le fine-tuning permet ainsi d'exploiter les connaissances préexistantes du modèle tout en l'orientant vers une performance optimale sur la nouvelle tâche définie.

Le choix de Camembert pour cette tâche repose sur sa capacité avérée à comprendre le français à un niveau nuancé, ce qui en fait une base solide pour notre fine-tuning.

### 3.2. Dataset et prétraitement

Le corpus pour cette étude est constitué de textes francophones, annotés par un sentiment positif (1) ou négatif (0), représentant une diversité de contextes et d'expressions sentimentales. À l'origine, le dataset comptait 1,526,724 entrées, avec une répartition équilibrée entre les sentiments positifs (49.5%) et négatifs (50.5%). En raison de contraintes de ressources, nous avons sélectionné un sous-ensemble de 1,000 entrées, veillant à maintenir une distribution quasi équilibrée. Cette sélection stratégique vise à équilibrer la représentativité du dataset avec les limitations pratiques, sans sacrifier significativement les performances potentielles du modèle. Le prétraitement a inclus la suppression des emojis, l'élimination des textes de moins de trois tokens, et la normalisation en minuscules, aboutissant à 981 entrées utilisables.

Une image contenant texte, capture d’écran, diagramme, Rectangle

Description générée automatiquement

Les statistiques post-nettoyage indiquent une répartition légèrement ajustée 51.1% de textes négatifs contre 48.9% de positifs. Ces étapes de prétraitement sont essentielles pour éliminer les variations non pertinentes et se concentrer sur le contenu textuel significatif.

Nous avons remarqué que la longueur des textes dans les 981 tweets selectionnées variait de 5 à 45 tokens, avec une moyenne de 18.43 tokens. Pour cela, la longueur maximale des séquences (MAX\_LEN) a été fixée à 47 tokens pour accommoder tous les tokens d'entrée plus deux tokens spéciaux nécessaires au modèle,

Ce choix a été dicté par le désir d'optimiser la performance du modèle tout en maintenant l'efficacité du calcul, permettant ainsi au modèle de traiter l'intégralité des données sans perte d'information significative.

### 3.3.Sélection et Justification des Hyperparamètres

Le fine-tuning du modèle Camembert pour l'analyse des sentiments s'est appuyé sur une évaluation systématique des hyperparamètres afin d'optimiser la performance sur un ensemble de 981 tweets. Cette démarche méthodique visait à affiner les paramètres du modèle en fonction de critères spécifiques à la tâche, en tenant compte des connaissances intrinsèques acquises lors de l'entraînement initial sur un large corpus de texte en français

Les hyperparamètres que nous avons ajustés dans nos expérimentations incluent :

Le taux d'apprentissage, ou Learning Rate (LR), joue un rôle déterminant dans le processus d'apprentissage, influençant directement la vitesse à laquelle notre modèle peut converger vers une solution optimale. Nous avons testé deux valeurs de LR, 0.00005 et 0.00002, sélectionnées pour leur capacité à équilibrer la progression rapide avec la préservation de la stabilité lors de la mise à jour des poids du réseau. Un LR trop élevé risque de mener à une convergence instable, tandis qu'un LR trop faible ralentit l'apprentissage, un équilibre soigneusement maintenu en alignement avec les meilleures pratiques de fine-tuning des modèles pré-entraînés.

Pour ce qui est de l'optimiseur, nous avons comparé les performances d'Adam et AdamW, chacun ayant des caractéristiques qui le rendent approprié pour le deep learning. Adam, réputé pour sa robustesse, et AdamW, qui ajoute une gestion raffinée du déclin de poids, ont été évalués pour leur efficacité à minimiser la fonction de perte. AdamW s'est avéré particulièrement prometteur pour sa capacité à atténuer le surajustement, une considération vitale lors de l'ajustement des modèles pré-entraînés.

En explorant les paramètres bêta, nous avons choisi les valeurs standards (0.9, 0.999) qui régulent le déclin exponentiel des moyennes mobiles des gradients passés. Ces valeurs aident à équilibrer l'influence des gradients antérieurs et à réduire les oscillations potentielles durant l'entraînement. L'epsilon a été fixé à 1e-8, une valeur ajoutée pour prévenir les divisions par zéro et maintenir la stabilité numérique lors de l'optimisation.

Le déclin de poids, ou Weight Decay, a été arrêté à 0.01, introduit comme un terme régularisateur visant à limiter la complexité du modèle en pénalisant les poids élevés. Cette valeur a été choisie pour offrir une régularisation significative sans entraver excessivement l'apprentissage.

Enfin, le scheduler du taux d'apprentissage est un outil essentiel pour ajuster le LR pendant l'entraînement et améliorer ainsi la convergence du modèle. Nous avons testé le scheduler 'get\_linear\_schedule\_with\_warmup', qui augmente progressivement le LR, et 'ExponentialLR', qui le réduit de manière exponentielle, pour déterminer leur efficacité dans l'affinement de la convergence du modèle aux différents stades de l'entraînement.

Chaque configuration a été soumise à trois cycles d'entraînement, ou époques, pour évaluer sa performance et garantir la fiabilité des résultats obtenus.

## 4. Résultats et discussion

### 4.1.Résultats d’entrainement :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Suite à une série d'expérimentations avec les configurations d'hyperparamètres mentionnées, nous avons observé des différences significatives dans la performance du modèle, évaluée en termes de précision et de courbes de perte d'entraînement et de validation. Les configurations utilisant AdamW avec un LR de 0.00005 et le scheduler 'get\_linear\_schedule\_with\_warmup' ont montré une performance optimale, avec une précision atteignant 82.2785%. Cette combinaison a démontré un équilibre idéal entre rapidité de convergence et qualité de généralisation, indiquant sa supériorité pour le fine-tuning de Camembert dans notre contexte d'analyse des sentiments.

### 4.2.Analyse des Courbes de Perte d'Entraînement et de Validation :

Une attention particulière a été accordée à l'évolution des pertes d'entraînement et de validation. Cette observation nous permet d'évaluer non seulement la performance du modèle mais aussi d'appréhender le comportement d'apprentissage au fil du temps. À travers l'analyse des courbes de perte associées à notre configuration d'hyperparamètres optimale sur trois époques, nous avons pu tirer des conclusions significatives sur l'efficacité de notre approche.

Une image contenant texte, capture d’écran, ligne, Tracé

Description générée automatiquement

Observations :

Décroissance des Pertes d'Entraînement : Une réduction progressive et cohérente de la perte d'entraînement a été constatée, symbolisée par une courbe bleue descendante. Ce phénomène indique un apprentissage effectif du modèle, qui parvient à minimiser la fonction de perte de façon continue, témoignant ainsi d'une amélioration graduelle des performances sur le dataset d'entraînement.

Convergence des Pertes de Validation : Parallèlement, la perte de validation, représentée par une ligne pointillée orange, suit également une trajectoire descendante, suggérant une bonne capacité de généralisation du modèle aux données non rencontrées durant l'entraînement. Plus remarquable encore est la stabilisation de cette perte de validation qui tend à converger vers la perte d'entraînement, signalant l'absence de surapprentissage.

Réduction de l'Écart entre les Pertes : L'écart réduit et décroissant entre les pertes d'entraînement et de validation au fil des époques souligne l'aptitude du modèle à appréhender des schémas généralisables plutôt que de mémoriser spécifiquement les données d'entraînement.

Implications :

Ces observations suggèrent que le modèle pourrait bénéficier de cycles d'entraînement supplémentaires, puisque la tendance actuelle de la perte de validation ne montre aucun signe d'augmentation — un indicateur courant de surapprentissage. De plus, la stabilisation observée de la perte de validation peut indiquer le moment optimal pour conclure l'entraînement, afin de préserver la généralisation sans tomber dans le surajustement.

La performance notable de cette configuration d'hyperparamètres spécifique établit une référence pour les expérimentations futures. Des investigations ultérieures pourraient explorer l'ajustement des hyperparamètres ou l'intégration de techniques de régularisation avancées, dans le but de réduire encore la perte de validation tout en maintenant l'équilibre avec la perte d'entraînement.

Les résultats obtenus renforcent la pertinence de notre sélection d'hyperparamètres pour le fine-tuning du modèle Camembert. La convergence stable observée entre les pertes d'entraînement et de validation, ainsi que la réduction de l'écart entre ces deux, illustrent efficacement la capacité du modèle à généraliser. Ces facteurs sont indicatifs du potentiel du modèle à délivrer des prédictions précises et fiables sur des données nouvelles, confirmant ainsi la validité de notre approche de fine-tuning pour l'analyse des sentiments.

### 4.3.Évaluation des Performances du Modèle

Nous avons évalué également ses performances en utilisant une matrice de confusion et diverses métriques clés, à savoir la précision, le rappel (recall) et le score F1 pour la classe positive (sentiments positifs). Les résultats sont les suivants :

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

Après fine-tuning, le modèle présente une capacité remarquable à identifier les sentiments positifs, comme en témoigne un rappel élevé de 0.9545, indiquant que 95.45% des cas réellement positifs ont été correctement identifiés dans le dataset. Cependant, la précision pour la classe positive se situe à 0.6364, ce qui signifie que seulement environ 63.64% des prédictions de sentiments positifs par le modèle étaient exactes. Cette performance est complétée par un score F1 de 0.7636 pour la classe positive, reflétant une bonne performance globale tout en soulignant la possibilité d'améliorer la précision.

La matrice de confusion révèle en détail les performances de prédiction du modèle avec 42 vrais positifs (VP) et 11 vrais négatifs (VN), montrant une compétence solide dans la prédiction des sentiments. Toutefois, le modèle a également produit 24 faux positifs (FP), où des sentiments négatifs ont été incorrectement classifiés comme positifs, et 2 faux négatifs (FN), où il a échoué à détecter des sentiments positifs. Ces résultats indiquent une exactitude globale de 67.09%, suggérant une performance supérieure à une sélection aléatoire mais laissant une marge significative pour l'amélioration, en particulier dans la réduction des faux positifs.

Le modèle démontre une propension à classifier les entrées comme positives, ce qui est avantageux pour minimiser les cas manqués de sentiments positifs, mais cette tendance se fait au détriment de la précision. Bien que le score F1 élevé pour la classe positive indique une gestion efficace de cette catégorie, l'absence de métriques détaillées pour la classe négative limite notre capacité à évaluer l'équilibre global entre les performances des deux classes.

Pour optimiser davantage la performance du modèle, il est crucial de se concentrer sur la diminution des faux positifs. Une telle amélioration pourrait accroître la précision sans compromettre significativement le rappel. Les stratégies potentielles incluent l'ajustement des seuils de classification et l'adoption de techniques de régularisation plus robustes, visant à établir un meilleur équilibre entre la précision et le rappel.

En effet, bien que le modèle excelle à détecter les sentiments positifs, son inclinaison vers de telles prédictions engendre un volume notable de faux positifs. Des améliorations ciblées sont nécessaires pour raffiner sa performance, en vue de parvenir à un équilibre optimal qui soutiendra l'application pratique de l'analyse des sentiments dans des scénarios réels.

## 5.Conclusion :

L'utilisation de l'optimiseur AdamW, couplée à un taux d'apprentissage finement ajusté à 0.00005, a été guidée par la théorie et la recherche de meilleures pratiques dans le domaine du fine-tuning. Ces choix se sont avérés judicieux, comme le démontre le rappel élevé pour la classe des sentiments positifs, signalant que le modèle est hautement compétent pour identifier la positivité dans le texte. Cependant, la précision plus faible et l'exactitude globale, qui est de 67.09%, suggèrent que la configuration actuelle peut être encore optimisée. En particulier, la tendance du modèle à prédire favorablement la classe positive appelle à une stratégie révisée pour réduire le nombre de faux positifs.