Analyse et visualisation de données

Rayan Leveque

Avril 2025

1 Introduction

Dans le cadre de ce mémoire, cette étude explore l'utilisation des grands modèles de langage (LLM) afin d'améliorer la qualité des transcriptions produites par la reconnaissance optique de caractères (OCR). Pour quantifier ces améliorations, la méthodologie proposée par Boros *et al.*[1] est reprise ; elle examine l'efficacité des modèles de langage dans la correction des erreurs de transcription de documents historiques, en comparant différentes architectures et stratégies.

1.1 Transformation : Post-traitement

La transformation des données extraites vise à affiner et structurer le texte reconnu. Ce post-traitement a pour objectif d'améliorer la qualité des résultats issus de l'OCR et d'en faciliter l'exploitation.

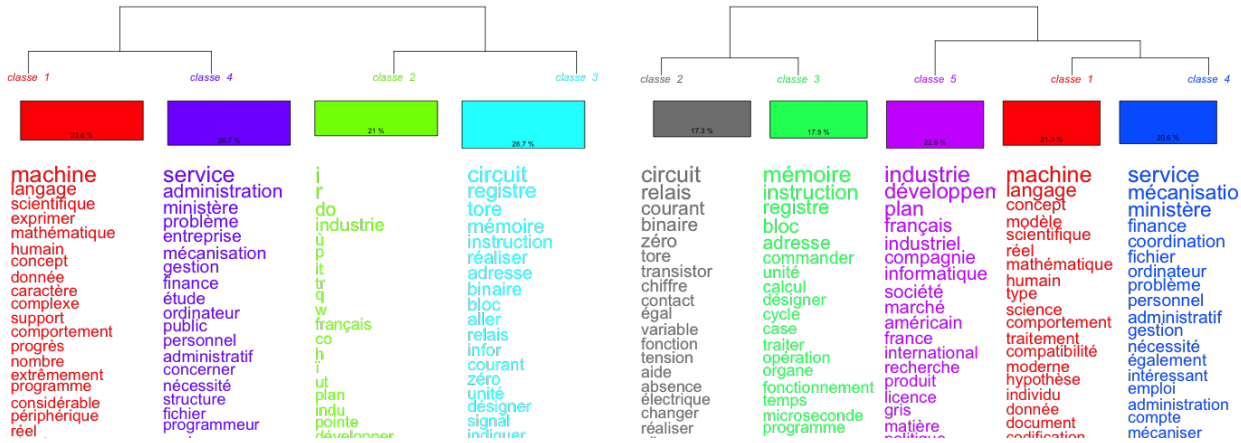
Pourquoi post-traiter ?

La question du post-traitement est évidente pour tout projet de traitement automatique des langues, l'objectif est de maximiser la qualité de *l'input* afin d'améliorer *l'output*[2]. à corriger Toutefois, l'étude *Quantifying the Impact of Dirty OCR on Historical Text Analysis*[3] montre que l'impact de la qualité d'un traitement OCR varie selon les méthodes d'analyse quantitative mobilisées. Les topic models, par exemple, se révèlent relativement robustes aux erreurs d'OCR, tandis qu'une approche co-occurrentielle se montre beaucoup plus sensible au bruit. La comparaison des résumés statistiques du corpus destiné à l'étude textométrique, détaillée ultérieurement, illustre l'intérêt d'un post-traitement. Dans la version brute, le nombre de formes est presque deux fois supérieur et la proportion d'hapax[4] augmente de 72% (voir Tableau 1). Lorsque le même algorithme est appliqué à un corpus traité ou non, les distorsions présentes font apparaître des résultats significativement différents : le nombre de classes issues d'une classification hiérarchique descendante (CHD)[5] ainsi que leur cohérence lexicale s'en trouvent altérés (voir Figure 1). Une classe dépourvue de cohérence sémantique émerge par exemple dans la version non-normalisée, tandis qu'elle disparaît après le traitement.

1. Boros et al., "Post-Correction of Historical Text Transcripts with Large Language Models".
2. *Garbage In Garbage Out*
3. Hill et Hengchen, "Quantifying the Impact of Dirty OCR on Historical Text Analysis".
4. Les formes n'apparaissant qu'une seule fois
5. Algorithme de classification hiérarchique descendante.

Résumé statistique	Corpus ocerisé	Corpus post-traité
Nombre d'occurrences	84 354	68 560
Nombre de formes	7 089	3 842
Nombre d'hapax	2 382	1 392

TABLE 1 – Résumé statistique des deux corpus étudiés avant et après post-traitement



Avant traitement Après traitement

FIGURE 1 – Comparaison des Classification Hiérarchique Descendante

Les dangers liés au post-traitementchanger de place

Meme si les résultats semblent plus cohérent et intelligible et que cela nous donne une plus grande impression de restitution d'un réel, nous devons garder en tête, comme discuté plus tôt dans se travail, des biais incontournables liés à l'utilisation de LLM à corriger

Choisir une méthode de post-traitement

Les méthodes de post-traitement mobilisables pour améliorer les résultats d'une reconnaissance optique de caractères peuvent être regroupées en deux grandes familles : les approches fondées sur des règles explicites et celles reposant sur l'apprentissage automatique.

Les méthodes à base de règles (ou *rule-based*) s'appuient sur des dictionnaires, des listes de formes connues, ou des expressions régulières pour corriger les erreurs les plus fréquentes. Faciles à mettre en œuvre et peu coûteuses en calcul, elles permettent un contrôle précis sur les transformations appliquées. Leur efficacité reste cependant limitée aux erreurs régulières ou prévisibles, et leur applicabilité se restreint souvent à un domaine ou un corpus spécifique.

Les approches d'apprentissage automatique, quant à elles, visent à apprendre les régularités à partir des données. Elles reposent sur des modèles capables de généraliser au-delà de règles explicites. Parmi elles, les réseaux neuronaux [6] se sont imposés ces dernières années comme des outils puissants pour le traitement automatique des langues. Leur capacité à

6. Notamment les architectures de type *Transformers*, introduites par l'article fondateur *Attention Is All You Need* (Vaswani *et al.*, 2017), à l'origine des modèles de langage de grande taille tels que les *Generative Pretrained Transformers* (GPT).

prendre en compte un contexte large, voire global, permet de traiter des ambiguïtés linguistiques ou des erreurs structurelles que des approches plus restreintes ne parviennent pas à résoudre. Ces méthodes sont plus flexibles, mais aussi plus exigeantes en ressources, et peuvent parfois introduire des corrections erronées ou imprévues.

Dans le cadre de ce mémoire, différentes stratégies ont été envisagées. Les méthodes fondées sur des règles se sont révélées insuffisantes compte tenu de la diversité des erreurs produites par l'OCR sur des documents anciens, souvent hétérogènes. Les approches de machine learning traditionnelles auraient nécessité la constitution d'un jeu de données annoté spécifiquement adapté, ce qui se heurte à l'extrême hétérogénéité du corpus étudié, tant sur le plan linguistique que structurel — et aurait représenté un coût trop élevé pour ce projet.

L'option retenue, fondée sur des LLMs, a été choisie pour sa souplesse d'application. Cette approche permet d'envisager une correction «par le sens», sans avoir à entraîner un modèle dédié.

Pourquoi utiliser des LLMs

Les LLMs représentent une avancée récente dans le domaine des réseaux neuronaux. Leur application au post-traitement OCR s'inscrit dans une dynamique plus large d'intégration du contexte sémantique dans les tâches de correction automatique.

Alors que les modèles traditionnels sont souvent limités à un contexte restreint, les LLMs ont la capacité de traiter de larges unités textuelles, incluant des paragraphes entiers. Cette profondeur contextuelle leur permet de corriger des incohérences sémantiques ou syntaxiques subtiles, qui passeraient inaperçues dans des systèmes plus simples.

Un autre avantage majeur réside dans leur polyvalence. En étant préentraînés sur de vastes corpus multilingues et hétérogènes, les LLMs peuvent être appliqués sans adaptation préalable à une grande diversité de textes, ce qui les rend particulièrement adaptés aux corpus historiques et bruités. Cette capacité à fonctionner en zéro-shot réduit considérablement les besoins en données annotées spécifiques, tout en assurant un haut niveau de performance.

Un manque de consensus

Les résultats des études récentes consacrées à l'emploi des LLMs pour la correction post-OCR sont partagées.

D'un côté, Zhang *et al.*[7] obtiennent des gains notables. Leur protocole commence par estimer la qualité du texte OCR, afin d'adapter dynamiquement la stratégie de correction ; la vaste fenêtre de contexte des LLM corrige alors des erreurs qui échappent aux méthodes traditionnelles basées sur des fenêtres textuelles glissantes.
Dans une perspective similaire, Thomas *et al.*[8] démontrent que l'utilisation de Llama 2 surpasse nettement BART pour la détection et la correction d'erreurs.
À l'inverse, Boros *et al.*[9] concluent que, dans leur cadre expérimental, les modèles de langage n'améliorent pas, voire dégradent fortement les transcriptions de documents historiques. Les auteurs soulignent que la variabilité typographique, la rareté des caractères anciens et les spécificités linguistiques constituent autant d'obstacles à la généralisation des LLM pour

7. Zhang et al., "Post-OCR Correction with OpenAI's GPT Models on Challenging English Prosody Texts".
8. Thomas, Gaizauskas et Lu, "Leveraging LLMs for Post-OCR Correction of Historical Newspapers".
9. Boros et al., "Post-Correction of Historical Text Transcripts with Large Language Models".

des tâches de corrections.
Dans cette perspective, une question demeure : dans quelle mesure les LLM peuvent-ils constituer une solution robuste et généralisable pour la correction post-OCR ? Pour explorer cette problématique, nous avons choisi d'adapter la méthodologie initialement développée par Boros *et al.*

Difficultés liées à la reprise d'une méthodologie

La réappropriation et l'adaptation de la méthodologie proposée ont présenté plusieurs difficultés. Un investissement conséquent en temps a été nécessaire pour la refactorisation des fonctions, notamment en raison des mises à jour apportées à la syntaxe des appels API, entre la rédaction initiale de l'article et la période actuelle. Par ailleurs, il a fallu intégrer la gestion du paramètre de température ainsi que la fonctionnalité de *role prompting*.

Cadre expérimental

Choix des modèles

Dans cette expérimentation, plusieurs modèles récents ont été mobilisés, mêlant modèles propriétaires et modèles ouverts afin d'évaluer leur performance. Deux familles principales ont été retenues : les modèles propriétaires d'OpenAI, et les modèles open source accessibles via la plateforme Novita. Le coût total de l'expérimentation s'est élevé à une vingtaine d'euros répartis entre ces deux plateformes. Les calculs effectués avec les modèles open-source n'ont consommé qu'environ 2 €, la différence résultant du dépôt minimal de 10 € exigé par la plateforme Novita.

- **Modèles GPT (propriétaires)** :
  - GPT-4.1 : Version avancée du modèle GPT-4 optimisé en précision et cohérence.
  - GPT-4.1-mini : Version allégée du GPT-4.1 destinée à un usage plus rapide tout en conservant une qualité élevée.
  - GPT-3.5-turbo : Ancien modèle d'OpenAI, désormais considéré comme obsolète par rapport aux versions plus récentes de GPT-4. Il offre des performances correctes pour des tâches simples, mais montre des limites marquées en termes de précision et de compréhension contextuelle..
- **Modèles ouverts (open source)** :
  - Deepseek-v3-turbo : Modèle optimisé par DeepSeek se distinguant par sa capacité d'inférence efficace tout en conservant une bonne qualité d'instruction.
  - Llama-4-maverick-17b-128e-instruct-fp8 : Modèle proposé par Meta, de taille intermédiaire et intensivement entraîné pour répondre efficacement aux instructions spécifiques.
  - Llama-4-scout-17b-16e-instruct : Version parallèle du modèle Llama de Meta, optimisée pour une génération robuste et une bonne généralisation.
  - Gemma-3-27b-it : Modèle ouvert de Google, de taille supérieure.

Préparation des données

Le corpus INA utilisé dans l'article a été récupéré ; il présente l'avantage d'être entièrement en français. Il s'agit de transcriptions audio d'extraits radiophoniques, comprenant des discours politiques, des programmes d'information et des émissions de divertissement.

Le second corpus est extrait d'un cours d'informatique donné à l'ENA en 1969 par Boucher. Afin de disposer d'une *vérité de terrain* pour l'évaluation, nous avons réalisé une transcription manuelle, six pages ont été saisies, puis découpées en 52 segments homogènes. L'ensemble représente 6 250 caractères au total.

Conception des prompts

Nous avons conceptualisé deux prompts offrant un guidage plus ou moins important [10] (voir Tableau 2). Chacun de ces prompts est décliné avec ou sans recours à la méthode du *role prompting*(respectivement *prompt_ complex* et *prompt_ basic*). La documentation d'OpenAI suggère de rapprocher les rôles *system* [11] et *user* à une fonction et ses arguments dans un langage de programmation. L'idée est ici de séparer plus distinctement le prompt de l'objet à traiter.

```
messages = [{"role": "user", "content": prompt}]
```

Sans role prompting

```
messages = [
    {"role": "system", "content": prompt},
    {"role": "user",   "content": text}
]
```

Role promptimg

Température

Dans nos expériences, nous avons testé deux valeurs pour le paramètre *temperature* : 0.05 et 0.7. Ce paramètre contrôle la créativité du modèle en ajustant la probabilité des réponses générées. À une température basse (*temperature = 0.05*), le modèle tend à produire des réponses plus déterministes et cohérentes, tandis qu'une température plus élevée (*temperature = 0.7*) introduit davantage de variabilité et de créativité dans les résultats. Cette distinction permet d'observer comment le modèle réagit.

Métrique d'évaluation

Afin d'évaluer objectivement la qualité des résultats obtenus, nous reprenons la métrique introduite dans l'article de Boros *et al.*, appelée PCIS [12], reposant sur la différence relative entre deux distances de Levenshtein. La distance de Levenshtein permet de mesurer quantitativement l'écart existant entre deux chaînes de caractères, en déterminant le nombre

10. *Few-shot* et *Zero-shot*, **touvronLLaMAOpenEfficient2023**
11. Rebaptisé récemment *developer* pour plus de clarté. https ://platform.openai.com/docs/guides/text ?api-mode=responses#message-roles-and-instruction-following
12. Post-Correction Improvement Score

Basic-1	Transcris le texte océrisé suivant (extrait d'une numérisation par OCR), en corrigeant les erreurs introduites par le processus d'OCR. Assure-toi de corriger tous les mots mal interprétés en restant le plus possible proche de la forme syntaxique et lexicale du texte. **Étapes** — Identifier les erreurs courantes dues à l'OCR : — Lettres mal reconnues (ex. « rn » → « m »). — Mauvaise segmentation ou caractères spéciaux erronés. — Ne pas changer le sens ni la syntaxe des phrases. **Format de sortie** Le texte transcrit correctement, paragraphe normal, même découpage que l'original. **Notes** — Attention aux caractères spéciaux et à la ponctuation mal reconnue. — Ne pas ajouter de texte avant ou après. — *Don't add any word that is not from the input.* **Exemple** *Entrée* : « Le secetariar générame du cnoseil q'ètait resopnsbile poru cela. » *Sortie* : « Le secrétariat général du conseil était responsable pour cela. » TEXTE : {{TEXT}}
Basic-2	Corrige le texte suivant, ne rajoute absolument rien de superflu. TEXTE : {{TEXT}}
Complex-1	Transcris un texte océrisé (extrait d'une numérisation par OCR), en corrigeant les erreurs introduites par le processus d'OCR. Assure-toi de corriger tous les mots mal interprétés en restant le plus possible proche de la forme syntaxique et lexicale du texte. **Étapes** — Lis attentivement le texte pour identifier les erreurs courantes dues à l'OCR : — Lettres mal reconnues (ex. « rn » interprété comme « m ») — Mauvaise segmentation des mots ou caractères spéciaux erronés — Ne modifie pas le sens ni l'organisation des phrases **Format de sortie** Le texte transcrit correctement, format de paragraphe normal, même découpage en paragraphes que le texte original. **Notes** — Attention aux caractères spéciaux ou à la ponctuation mal interprétés. — Ne rajoute pas de texte avant ou après. *Don't add any words that are not from the input.* **Exemple** Entrée : « Le secetariar générame du cnoseil q'ètait resopnsbile poru cela. » Sortie : « Le secrétariat général du conseil était responsable pour cela. »
Complex-2	Corrige le texte suivant, ne rajoute absolument rien de superflu.

TABLE 2 – Prompt templates (Basic-1, Basic-2, Complex-1 et Complex-2).

```
messages = [
    {
        "role": "system",
        "content": (
            "Corrige le texte suivant,
    ne rajoute absolument rien de
    superflu.\n\n
        TEXTE : {Un tel traduzteur, qui
    nous apparait aujourd'hui
    d'écriture élémentaire, nécessite
    quelques centaines d'instructionsy
    et d'autant plus, évidemment, que
    l'on a souhaité plus de commodités
    dans le langage externe.[...]}"
        )
    }
]
```

Exemple `prompt_basic_02`

```
messages = [
    {
        "role": "system",
        "content": "Corrige le texte
    suivant, ne rajoute absolument rien
    de superflu."
    },
    {
        "role": "user",
        "content": (
        "Un tel traduzteur, qui nous
    apparait aujourd'hui d'écriture é
    lémentaire, nécessite quelques
    centaines d'instructionsy et
    d'autant plus, évidemment, que l'on
    a souhaité plus de commodités dans
    le langage externe.[...]"
        )
    }
]
```

Exemple `prompt_complex_02`

minimal d'opérations élémentaires (ajout, suppression ou substitution de caractères) nécessaires pour transformer une chaîne en une autre. Dans le contexte de notre étude, cette métrique est appliquée afin de comparer quantitativement les résultats des transcriptions obtenues avant et après correction assistée par LLM. Cependant, appliqué tel quel, le PCIS pousse à une interprétation erronée des résultats. Les auteurs précisent que « *The improvement score ranges from* -1 *to* 1 *: negative values indicate deterioration, positive values indicate improvement, and* 0 *indicates no change.* » Or, le texte produit par l'OCR n'est jamais entièrement incorrect. Dans notre corpus par exemple, le score moyen avant correction est déjà de 0,84. L'espace de variation est donc asymétrique, borné entre $-0{,}84$ (détérioration totale) et $+0{,}16$ (amélioration parfaite). Employer l'intervalle $[-1, 1]$, surestime la plage d'amélioration possible et masque la difficulté d'accroître la qualité d'un texte déjà partiellement correct.

Résultats et discussion

Les cartes de chaleur (Figure 2, Figure 3, Figure 4, Figure 5) présentent le PCIS en fonction du corpus et de la température en abscisse et des modèles utilisés en ordonnée en fonction des différents prompts.

Les cartes de chaleur mettent en évidence les éléments suivants :

— La température de 0,05 donne de meilleurs résultats que la température de 0,7.

— Les modèles GPT-4 présentent des performances supérieures aux modèles ouverts, quels que soient les prompts, les températures ou les corpus considérés.

— Les prompts utilisant le *role prompting* obtiennent de meilleurs résultats (Prompt_Complex-01 et Prompt_Complex-02)

— Les prompts utilisant le *few-shot prompting* obtiennent de meilleurs résultats (Prompt_Basic-01 et Prompt_Complex-01)

— Pour le corpus *ina*, la reproduction des résultats a été confirmée avec un score identique de 0,02 pour le modèle GPT-4 (voir **??**).reformule

Une des explications des mauvais résultats des modèles Llama est la présence, en début ou en fin de correction, d'éléments conversationnels superflus. Les résultats présentés dans les tableaux (Tableau 3, Tableau 4) montrent que l'utilisation de LLMs pour des tâches de correction post-OCR permet d'améliorer ou de conserver la qualité des transcriptions jusqu'à 75%, dans le cas du modèle le plus performant.
Une analyse plus fine des facteurs spécifiques entraînant la dégradation des transcriptions pourrait considérablement accroître cette performance globale.

En définitive, les résultats indiquent clairement que les LLMs, particulièrement les modèles GPT-4 associés à des techniques de *few-shot prompting* et de *role prompting*, permettent d'atteindre des niveaux de performance satisfaisants pour la correction post-OCR, sans nécessiter de procédures complexes.

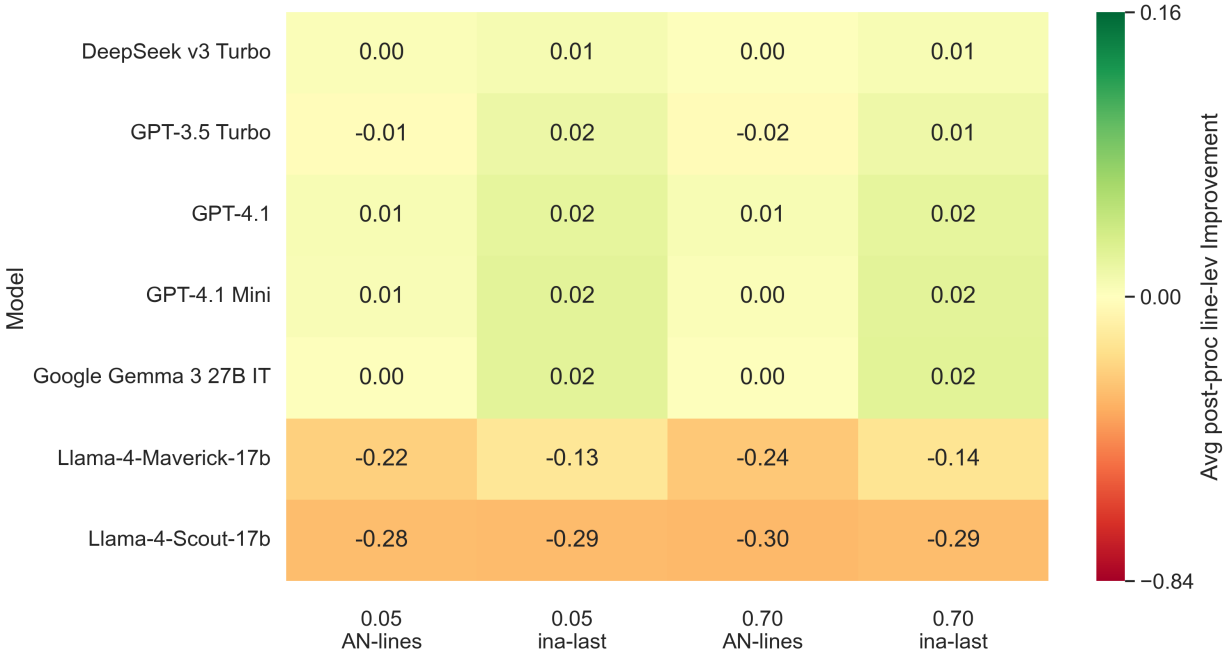


FIGURE 2 – Amélioration moyenne de la distance de Levenshtein par ligne selon les modèles et les paramètres de température (Prompt Basic-01)

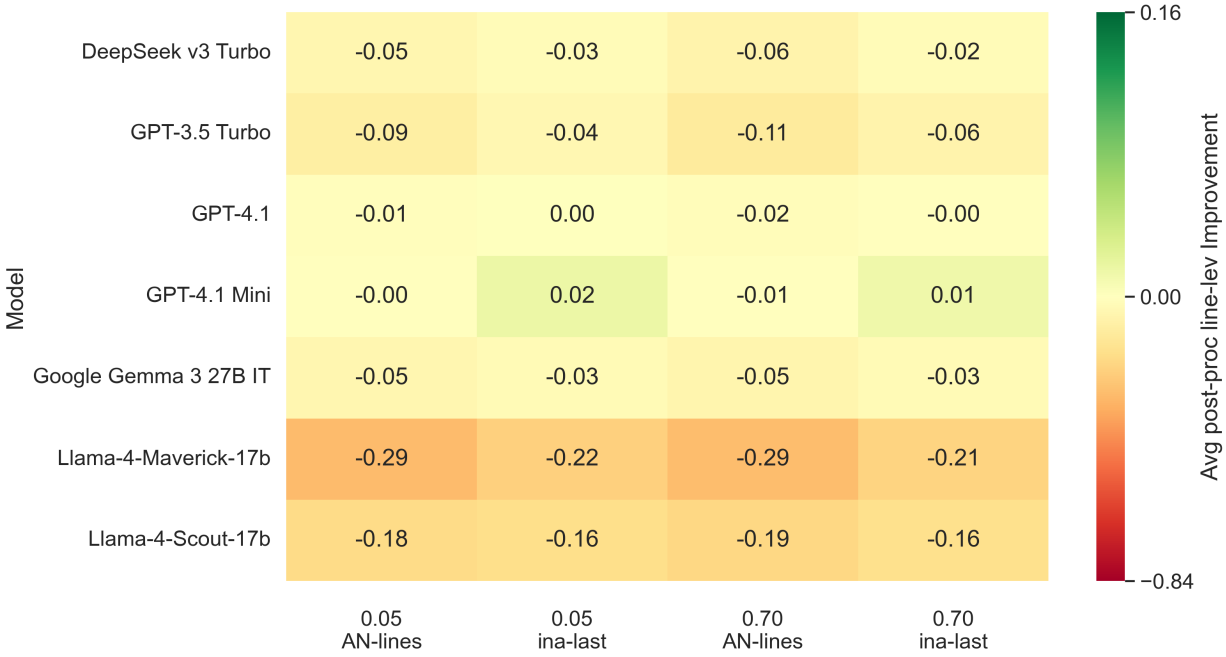


FIGURE 3 – Amélioration moyenne de la distance de Levenshtein par ligne selon les modèles et les paramètres de température (Prompt Basic-02)

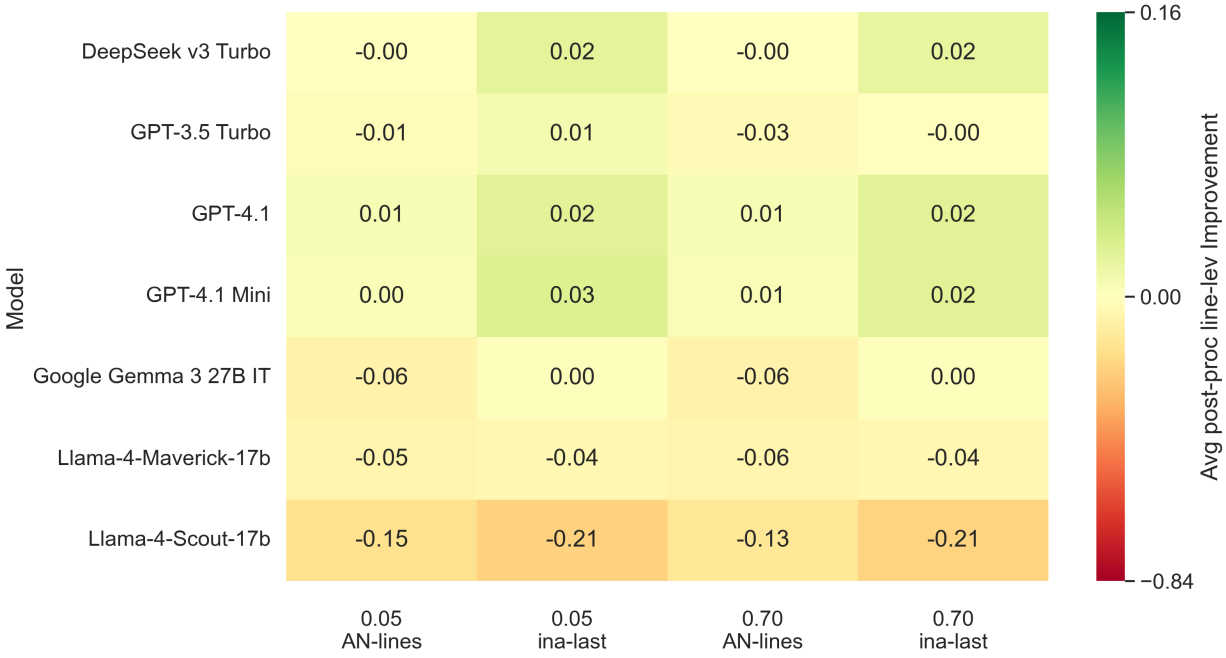


FIGURE 4 – Amélioration moyenne de la distance de Levenshtein par ligne selon les modèles et les paramètres de température (Prompt Complex-01)

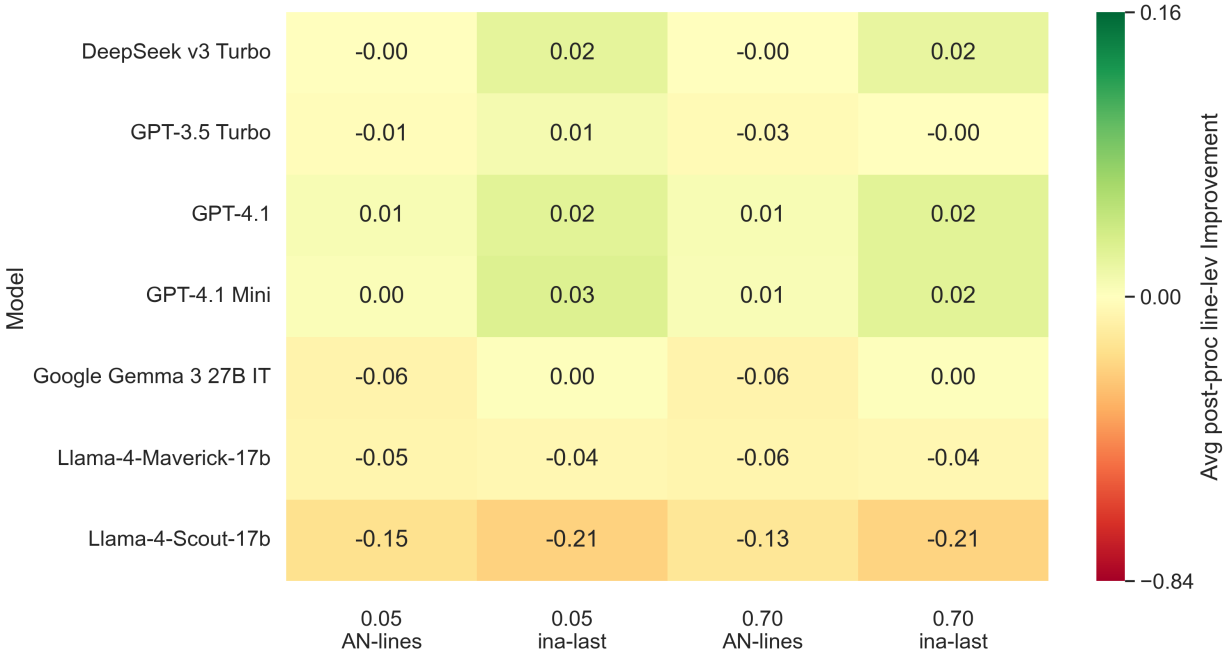


FIGURE 5 – Amélioration moyenne de la distance de Levenshtein par ligne selon les modèles et les paramètres de température (Prompt Complex-02)

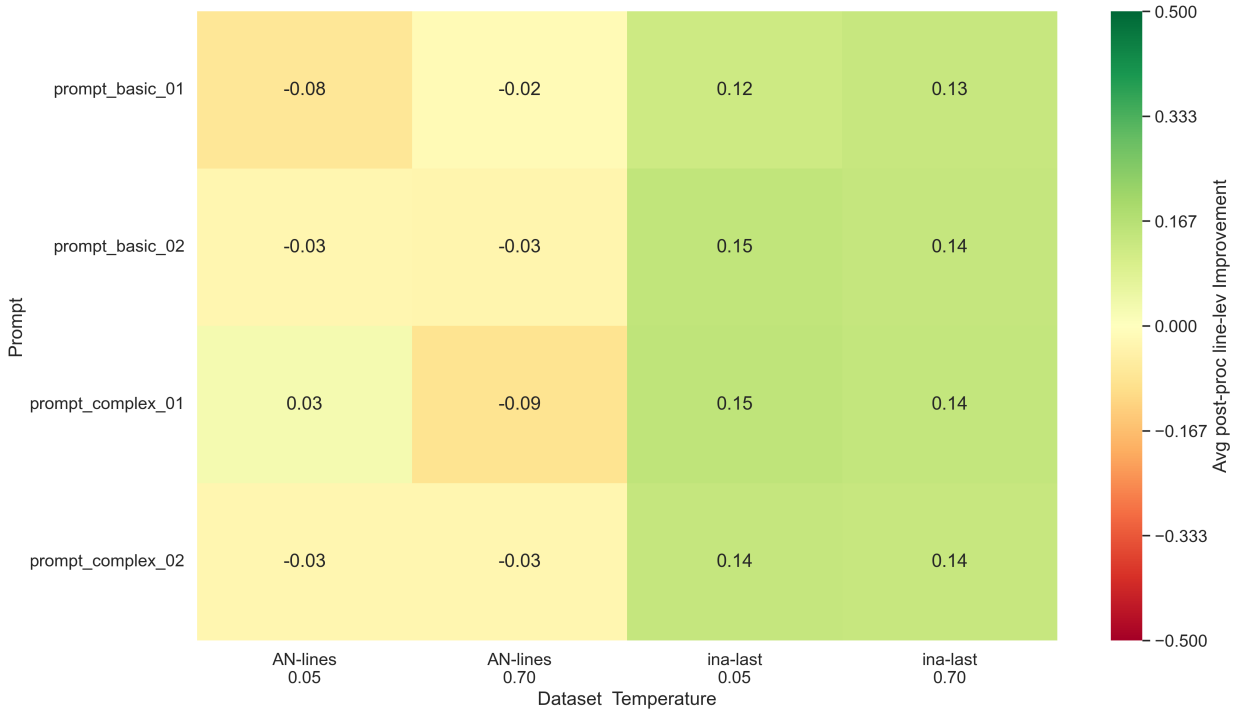


FIGURE 6 – Amélioration du WER selon le prompt, température et jeu de données pour le modèle GPT-4.1-mini

Bande d'amélioration modèle	Dégradation	Aucun changement	Amélioration
gpt-4.1-mini	1060	402	2857
gpt-4.1	1279	336	2703
deepseek-deepseek-v3-turbo	1715	304	2299
google-gemma-3-27b-it	1693	325	2301
gpt-3.5-turbo	1983	203	2133
llama-4-maverick-17b	2475	94	1748
llama-4-scout-17b-16e	3946	53	319

TABLE 3 – Bandes d'amélioration pour différents modèles

Bande d'amélioration (%) modèle	Dégradation	Aucun changement	Amélioration
gpt-4.1-mini	24.542718	9.307710	66.149572
gpt-4.1	29.620195	7.781380	62.598425
deepseek-v3-turbo	39.717462	7.040296	53.242242
google-gemma-3-27b-it	39.198889	7.524890	53.276221
gpt-3.5-turbo	45.913406	4.700162	49.386432
llama-4-maverick-17b-128e	57.331480	2.177438	40.491082
llama-4-scout-17b-16e	91.384900	1.227420	7.387679

TABLE 4 – Bandes d'amélioration (en pourcentage) pour différents modèles