| | |
|---|---|
| **Titre:** Title: | Modèles de langage au service de la recommandation basée sur des sessions |
| **Auteur:** Author: | Anis Redjdal |
| **Date:** | 2024 |
| **Type:** | Mémoire ou thèse / Dissertation or Thesis |
| **Référence:** Citation: | Redjdal, A. (2024). Modèles de langage au service de la recommandation basée sur des sessions [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/59177/ |

**POLYTECHNIQUE MONTRÉAL**
affiliée à l'Université de Montréal

# Modèles de langage au service de la recommandation basée sur des sessions

**ANIS REDJDAL**
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Août 2024

**POLYTECHNIQUE MONTRÉAL**
affiliée à l'Université de Montréal

Ce mémoire intitulé :

**Modèles de langage au service de la recommandation basée sur des sessions**

présenté par **Anis REDJDAL**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

**Omar ABDUL WAHAB**, président
**Michel DESMARAIS**, membre et directeur de recherche
**Zohreh SHARAFI**, membre

# DÉDICACE

*Je souhaite dédier ce mémoire à plusieurs personnes sans qui ce travail n'aurait pas été possible.*

*Tout d'abord, je tiens à exprimer ma profonde gratitude à mon directeur de recherche, Michel Desmarais, pour son encadrement exceptionnel, ses conseils avisés et son soutien tout au long de ce projet. Ses connaissances approfondies et son approche pédagogique ont été inestimables pour la réalisation de ce travail.*

*Je tiens également à remercier chaleureusement Luis Pinto, scientifique de données remarquable avec qui j'ai eu le privilège de collaborer durant le partenariat entre mon école, Polytechnique, et Mitacs. Sa passion pour la recherche et son expertise ont grandement contribué à enrichir ce mémoire.*

*À ma compagne, merci pour ton soutien et ta compréhension tout au long de ce projet, de croire en moi et de m'accompagner chaque jour.*

*À mes amis et à ma famille, je dédie également ce travail. Votre soutien constant, vos encouragements et votre compréhension m'ont donné la force et la motivation nécessaires pour mener à bien ce projet. Votre présence à mes côtés a été une source de réconfort et d'inspiration.*

*Merci à tous....*

## REMERCIEMENTS

J'aimerais remercier mon directeur de recherche Michel Desmarais ainsi que le scientifique de données d'Air Liquide Luis Pinto qui m'ont accompagné dans la réflexion et la réalisation de mes travaux.

# RÉSUMÉ

La recommandation basée sur les sessions (SBR) utilise le comportement passé de l'utilisateur au sein d'une session unique pour établir la prédiction de ses actions futures et effectuer des recommandations. Une session représente une série d'interactions consécutives d'un utilisateur avec des articles (ou *items*) dans un contexte donné. Par exemple, sur un site de commerce électronique, une session peut commencer lorsque l'utilisateur se connecte et se termine lorsqu'il quitte le site, les données de cette session comprennent les articles qu'il a consultés dans un ordre précis. Comprendre et modéliser ces sessions est essentiel pour améliorer la pertinence des recommandations et offrir une expérience utilisateur personnalisée et engageante.

Diverses méthodes existent, toutes centrées sur la modélisation de sessions d'actions et la prédiction d'actions futures. Depuis 2019, les systèmes SBR basés sur les *Graph neural networks* (GNN) ont été largement explorés et se sont imposés comme l'approche de prédilection pour la SBR, notamment dans les tâches de prédiction du prochain article. Dans cette étude, nous proposons une méthode basée sur des modèles de langage Transformer pour réaliser la prédiction du prochain article dans la SBR. En nous concentrant sur les architectures des modèles de langage les plus populaires et les plus performants, nous utilisons ces modèles Transformer pour la tâche de prédiction du prochain article (qui s'apparente à la tâche de prédiction du prochain mot dans le traitement du langage naturel (NLP)) et les comparons aux modèles basés sur les GNN. Nous discutons des avantages uniques que chaque modèle offre aux systèmes SBR.

Afin d'utiliser des architectures de modèles de langage basés sur les Transformers, on introduit dans cette étude une nouvelle technique de masquage adaptée aux systèmes de recommandation SBR. Nous appelons cette technique *sequential masked modeling (SMM)*.

Dans un premier temps, nous évaluons l'efficacité de notre nouvelle approche de masquage pour les Transformers dans le domaine de la SBR en la comparant aux techniques de masquage précédemment utilisées dans le NLP comme *masked language modeling* (MLM) et *causal language modeling* (CLM). Les résultats démontrent que notre nouvelle technique de masquage améliore grandement le score de prédiction ainsi que la vitesse d'entrainement des Transformers dans le domaine de la SBR. De plus, nos recherches consistent à montrer que si les modèles Transformer sont améliorés et optimisés, comme l'ont été les GNN ces dernières années, ils peuvent alors rivaliser avec ces derniers pour des tâches de SBR. Nous intégrons donc des techniques récentes de modèles de langage larges (LLM) à nos architectures de mo-

dèles Transformer, notamment celles de Llama 3, et démontrons des gains significatifs. Ces travaux confirment le potentiel des modèles Transformer dans le domaine de la SBR.

Avec l'implémentation de trois architectures de Transformers populaires (BERT, GPT, et DeBERTa) optimisées à l'aide de notre nouvelle approche de masquage ainsi que d'autres techniques d'optimisation provenant des LLM récents, les résultats montrent que ces trois architectures offrent des performances supérieures à celles d'un GNN de base sur les trois jeux de données utilisés pour l'évaluation. De plus, les résultats montrent que notre approche de masquage n'est efficace que sur les architecutres de type encodeur qui sont BERT et DeBERTa. L'architecture BERT se révèle légèrement supérieure à celle de DeBERTa.

Finalement, nous cherchons à démontrer que les modèles Transformer appliqués à la SBR peuvent être pré-entraînés avec des données non-ordonnées afin d'améliorer leur performance avec des données ordonnées (les deux ensembles de données doivent provenir de distributions différentes idéalement). En effet, de nombreuses entreprises possèdent des données non-ordonnées qui ne sont pas utilisées pour l'entraînement de modèles de systèmes de recommandations SBR car l'ordonnancement des séquences est crucial pour la tâche de prédiction du prochain article. Ainsi, les modèles de langage peuvent être pré-entraînés sur ces données non-ordonnées puis affinés avec les données ordonnées pour obtenir de meilleures performances.

Pour le pré-entraînement, nous utilisons deux jeux de données : le premier constitué de séquences non-ordonnées et le deuxième de séquences ordonnées. Ces jeux de données appartiennent à l'entreprise Air Liquide Canada (ALC). Dès lors, nous obtenons des résultats significativement meilleurs sur la version ordonnée des données lorsqu'on effectue une première étape de pré-entraînement avec les données non-ordonnées. Les résultats de l'affinage avec différentes architectures de Transformers montrent que DeBERTa donne les meilleurs résultats parmi les trois lorsqu'il s'agit de pré-entraînement. De plus, nous simulons un jeu de données non-ordonnées sur la partie de Yoochoose que nous n'utilisons pas pour l'évaluation afin d'effectuer un pré-entraînement sur cette partie. Ces résultats surpassent même la performance du meilleur GNN à ce jour avec le jeu de données Yoochoose.

Les jeux de données Yoochoose, Diginetica et Tmall ainsi que le code sont disponibles publiquement à l'adresse https ://github.com/anisredjdal.

En conclusion, avec l'introduction de notre nouvelle technique de masquage SMM couplée à d'autres optimisations, les modèles Transformer s'annoncent prometteurs pour la recommandation basée sur les sessions, tout comme les GNN l'ont été ces dernières années. De plus, ces modèles peuvent bénéficier d'un pré-entraînement en cas de disponibilité de données provenant d'une autre distribution afin d'optimiser davantage leurs performances.

# ABSTRACT

SBR leverages a user's past behavior within a single session to predict their future actions and provide recommendations. A session represents a series of consecutive interactions between a user and items in a given context. For instance, on an e-commerce website, a session might begin when a user logs in and ends when they leave the site, with session data including the specific sequence of items viewed. Understanding and modeling these sessions is crucial for enhancing recommendation relevance and delivering a personalized and engaging user experience.

Various methods exist, all focused on modeling session actions and predicting future actions. Since 2019, GNN-based SBR systems have been extensively explored and have become the preferred approach for SBR, particularly in next-item prediction tasks. In this study, we propose a method based on Transformer language models to perform next-item prediction in SBR. By focusing on the most popular and effective language model architectures, we use these Transformer models for next-item prediction (similar to next-word prediction in NLP) and compare them to GNN-based models. We discuss the unique advantages each model offers to SBR systems.

To utilize Transformer-based language model architectures, we introduce a new masking technique tailored to SBR recommendation systems in this study, which we call *SMM*.

First, we evaluate the effectiveness of our new masking approach for Transformers in the SBR domain by comparing it to previously used masking techniques in NLP, such as MLM and CLM. The results show that our new masking technique significantly improves prediction scores as well as the training speed of Transformers in the SBR domain. Additionally, our research aims to demonstrate that if Transformer models are enhanced and optimized, as GNNs have been in recent years, they can then compete with GNNs for SBR tasks. We thus incorporate recent LLM techniques into our Transformer model architectures, including those from Llama 3, and demonstrate significant gains. This work confirms the potential of Transformer models in the SBR domain.

With the implementation of three popular Transformer architectures (BERT, GPT, and DeBERTa) optimized with our new masking approach as well as other optimization techniques from recent LLMs, the results show that these three architectures outperform a basic GNN on the three datasets used for evaluation. Moreover, the results show that our masking approach is effective only on encoder-type architectures, which are BERT and DeBERTa. The BERT architecture proves to be slightly superior to DeBERTa.

Finally, we seek to demonstrate that Transformer models applied to SBR can be pre-trained with unordered data to improve their performance with ordered data (ideally, both datasets should come from different distributions). Indeed, many companies possess unordered data that are not used for training SBR recommendation system models because the sequencing of events is crucial for next-item prediction tasks. Thus, language models can be pre-trained on these unordered data and then fine-tuned with ordered data to achieve better performance.

For pre-training, we use two datasets : the first consisting of unordered sequences and the second of ordered sequences. These datasets belong to the ALC company. As a result, we obtain significantly better results on the ordered version of the data when pre-training is first conducted with the unordered data. The fine-tuning results with different Transformer architectures show that DeBERTa delivers the best results among the three for pre-training. Additionally, we simulate an unordered dataset on the portion of Yoochoose that we do not use for evaluation to perform pre-training on this portion. These results even surpass the performance of the best GNN to date on the Yoochoose dataset.

The Yoochoose, Diginetica, and Tmall datasets, as well as the code, are publicly available at `https://github.com/anisredjdal`.

In conclusion, with the introduction of our new SMM masking technique coupled with other optimizations, Transformer models appear promising for session-based recommendation, much like GNNs have been in recent years. Additionally, these models can benefit from pre-training when data from another distribution is available to further optimize their performance.

# TABLE DES MATIÈRES

# LISTE DES TABLEAUX

# LISTE DES FIGURES

# LISTE DES SIGLES ET ABRÉVIATIONS

| | |
|---|---|
| SBR | Recommandation basées sur les sessions |
| MLM | Masked Language Modeling |
| GNN | Graph Neural Networks |
| RNN | Recurrent Neural Network |
| NLP | Traitement du Langage Naturel |
| LLM | Grands modèles de langage |
| RoPE | Rotation Positional Encoding |
| MoE | Mixture of Experts |
| MRR | Rang réciproque moyen |
| CNN | Réseaux de Neurones Convolutifs |
| SMM | Sequential Masked Modeling |
| ALC | Air Liquide Canada |
| KAN | Kolmogorov–Arnold Networks |
| CLM | Causal Language Modeling |
| RPE | Encodage de position inversé |
| CoPE | Contextual Position Encoding |
| PLM | Permutative Language Modeling |

# CHAPITRE 1 INTRODUCTION

Alors que l'intelligence artificielle continue de s'étendre à un rythme sans précédent, le déluge d'informations disponibles nécessite le déploiement de systèmes de recommandation de plus en plus sophistiqués. Ces outils sont cruciaux pour aider les utilisateurs à naviguer dans l'immensité du contenu, améliorant leur expérience sur diverses plateformes en ligne telles que les moteurs de recherche, le shopping en ligne et les applications de vidéos (comme tiktok et youtube). Les systèmes de recommandation traditionnels [8] dépendent généralement d'un historique des interactions et des préférences de l'utilisateur. Cependant, cette approche rencontre des limites dans les scénarios où les identités des utilisateurs sont partiellement connues ou totalement anonymes, se basant uniquement sur les actions effectuées au sein d'une seule session de courte ou longue durée. Dans ces situations, il devient essentiel d'interpréter adroitement les données comportementales séquentielles disponibles de ces sessions indépendantes pour formuler des recommandations efficaces. Cette nécessité souligne un défi significatif pour les modèles traditionnels, qui dépendent fortement des historiques d'interaction utilisateur-article pour produire des recommandations précises, et qui peinent souvent à s'adapter aux contraintes des scénarios basés sur les sessions (basés sur des séquences ordonnées et de tailles variées d'utilisateurs pouvant être anonymes). Pour cela, on considère deux sessions d'un même utilisateur complètement indépendantes.

L'intérêt pour les systèmes de recommandation SBR a connu une hausse notable. Dans cette approche, une session capture une séquence de comportements interactifs, tels que des clics dans des environnements de commerce électronique, sur une courte période. L'objectif de la SBR est de prévoir le prochain article qui pourrait intéresser un utilisateur, en se basant uniquement sur des séquences de données ordonnées anonymes et à court terme.

Le premier essor d'approches efficaces dans le domaine de la SBR a commencé à apparaître vers 2016, avec les RNN devenant la technique de prédilection pour les tâches de recommandation basées sur les sessions. L'étude [9] a initialement introduit une stratégie basée sur les RNN, qui a été ensuite affinée par l'augmentation des données et en tenant compte des déplacements temporels dans le comportement des utilisateurs, comme exploré par [10]. De plus, NARM [11] a introduit un système de recommandation RNN à double approche qui capture à la fois le comportement séquentiel des utilisateurs et leurs intentions principales. De même, STAMP [12] vise à identifier les préférences générales des utilisateurs et leurs intérêts actuels en employant de simples réseaux de neurones multicouches (MLP) intégrés avec un mécanisme d'attention. Malgré les défis naturellement associés aux RNN, tels

que la difficulté de représenter les utilisateurs avec des données limitées ou la complexité de modéliser les transitions lointaines d'articles dans une séquence [11, 12], les chercheurs ont progressivement amélioré cette méthodologie au fil du temps mais les défis associés restent bien présents.

Pour relever ces défis, l'introduction des GNN en 2019 a marqué une avancée significative dans les systèmes de recommandation SBR. Les GNN adaptés aux sessions [13] sont spécifiquement conçus pour générer des représentations pour les graphes, qui sont particulièrement aptes à naviguer les transitions complexes entre les données qui composent une séquence pour produire des vecteurs latents précis pour représenter ces mêmes séquences. Les GNN excellent dans la capture des transitions de données complexes que les méthodes séquentielles conventionnelles, telles que les approches basées sur les Chaînes de Markov (MC) [14] et les RNN, échouent souvent à découvrir. En générant des vecteurs d'incorporation d'articles précis, le modèle SR-GNN [13] proposé est capable de construire des représentations de session plus robustes, améliorant ainsi la précision de la prédiction pour la tâche de prédiction du prochain clic. Par la suite, les GNN sont devenus la méthodologie de premier plan dans le domaine de la SBR, surpassant régulièrement les approches précédentes dans les tâches de prédiction du prochain clic avec des métriques de performance toujours plus élevées. Le tournant s'est amorcé en 2019 avec l'introduction d'un modèle GNN standard [13]. Depuis, des avancées importantes ont été réalisées, comme le montrent les recherches ultérieures [15], [16], et [17].

En considérant la tâche de prédire le prochain article, cela nous a rappelé les défis associés à la prédiction du prochain mot dans le NLP. Ce parallèle nous a poussé à explorer l'application des modèles Transformer [2], en nous concentrant particulièrement sur les architectures de modèles de langage. Nous utilisons ces dernières afin de traiter des séquences ordonnées d'articles ou de clics plutôt que des représentations de mots dans une phrase. Nous proposons donc un parallèle entre les tâches de prédiction SBR et de NLP. Pour cette étude, nous avons développé des implémentations complètes des architecures de Transformer BERT, GPT-2 [18], DeBERTa [19] et Mamba [20] pour évaluer leurs performances sur la tâche de prédiction du prochain article. L'implémentation de ces modèles "à partir de zéro" était essentielle pour assurer l'absence de pré-entraînement, permettant une compatibilité avec notre *tokeniseur* de données (objets, clics,...) personnalisé et non un *tokeniser* de mots comme on les connait. Ce *tokeniseur* a été spécifiquement entraîné sur notre ensemble de données d'articles de e-commerce, facilitant une approche sur mesure pour comprendre les modèles d'interaction des utilisateurs anynymes.

Finalement, nous avons découvert à l'aide de l'entente *Mitacs* avec ALC qu'il était possible

d'utiliser des sessions de données non-ordonnées pour pré-entrainer nos architectures de modèles de lanagage. En effet, comme beaucoup de compagnies industrielles, *Air Liquide* possède des données non-ordonnées provenant des achats multiples en magasin (nous ne pouvons pas savoir quel article le client a séléctionné en premier au moment où il passe à la caisse) ainsi que des données ordonnées provenant des consultation d'articles des clients du site internet (nous pouvons savoir dans quel ordre le client a consulté les articles en analysant les clics effectués). Les données en magasin sont donc inutilisables pour du SBR car celles-ci ne se prêtent pas à la tâche. Néanmoins, les Transformers peuvent être pré-entrainées avec ces données et cela aide à une compréhension plus large de la distribution de ces dernières. Le pré-entrainement se fait donc à partir d'une distribution de données différente de l'affinage, néanmoins le vocabulaire d'articles à recommender par le modèle est le même pour les deux distributions c'est donc bien faisable. En plus des données de l'entreprise Air Liquide, nous montreront à l'aide du dataset Yoochoose [21] que ce pré-entrainement sur des données non-ordonnées couplé à un affinage sur des données ordonnées peut améliorer les performances de notre modèle de langage.

## 1.1 Éléments de la problématique

Avec l'essor des systèmes de recommandation, la capacité à fournir des suggestions personnalisées en temps réel est devenue essentielle pour améliorer l'expérience utilisateur sur les plateformes numériques. Cependant, cette tâche est particulièrement complexe dans des scénarios où les utilisateurs sont anonymes et où aucune donnée historique n'est disponible. Ces défis soulèvent plusieurs problématiques clés qui méritent une attention particulière.

### 1.1.1 Limitations des méthodes traditionnelles

Les systèmes de recommandation traditionnels se basent principalement sur l'historique des interactions utilisateurs pour formuler des suggestions pertinentes. Cette approche, bien qu'efficace dans des contextes où les données historiques sont abondantes, se heurte à des limitations significatives dans les situations où les utilisateurs sont nouveaux ou anonymes. Dans ces cas, il n'existe pas de données antérieures pour guider les recommandations, ce qui rend les modèles traditionnels inefficaces.

De plus, les modèles traditionnels ont tendance à mal gérer les interactions à court terme et séquentielles. Les méthodes basées sur les statistiques et les algorithmes de filtrage collaboratif [22] ne sont pas adaptées à capturer les dynamiques complexes des interactions utilisateur-article qui se produisent dans une seule session relativement courte. Par conséquent, il est crucial de développer des approches capables de s'adapter à ces environnements à court terme.

### 1.1.2 Défis des modèles basés sur les sessions

Les systèmes de recommandation SBR sont conçus pour surmonter les limitations des approches traditionnelles en se concentrant sur les interactions séquentielles au sein d'une seule session utilisateur. Cependant, cette approche présente également des défis uniques :

— **Capture de la dynamique séquentielle** : La modélisation des séquences d'interactions utilisateur-article nécessite des techniques avancées capables de capturer la temporalité et les transitions complexes entre les articles. Les RNN ont été largement utilisés à cette fin, mais ils souffrent de limitations en termes de capacité à gérer des dépendances à long terme et à représenter des utilisateurs avec des données limitées.

— **Modélisation des intentions utilisateur** : Une autre difficulté majeure est la compréhension des intentions principales des utilisateurs à partir de leurs interactions séquentielles. La prédiction des prochaines interactions implique que les modèles doivent saisir les objectifs sous-jacents des utilisateurs, ce qui nécessite des techniques de modélisation sophistiquées comme les mécanismes d'attention.

— **Optimisation de la taille des modèles** : Les modèles SBR, en particulier ceux basés sur des réseaux neuronaux complexes, peuvent être très gourmands en ressources de calcul. Il est donc essentiel de trouver un équilibre entre précision des recommandations et efficacité des calculs pour garantir des performances optimales en temps réel. De plus, la taille des séquences peut varier et certaines peuvent être trop longues pour chaque modèle qui possède une taille maximum fixe de séquences d'entrée. Si la taille maximale admissible est trop grande alors la complexitée des calculs sera décuplée si celle-ci est trop courte le modèle peu passer à coté de données importes il est donc important de trouver le bon équilibre.

### 1.1.3 Évolution des modèles et méthodologies

Face aux limitations des RNN, l'introduction des GNN a marqué une avancée significative dans les systèmes de recommandation basés sur les sessions. Les GNN sont particulièrement efficaces pour modéliser les relations complexes entre les articles en représentant les interactions sous forme de graphes. Cette approche permet de capturer plus finement les transitions et les dépendances entre les articles, améliorant ainsi la précision des recommandations.

En parallèle, l'application des modèles Transformer dans le domaine de la recommandation basée sur les sessions n'a pas été suffisamment exploitée dans le sens où uniquement des Tranformers de base ont été implémentés. Par exemple, la librairie "transformer4rec" développée par Méta et Nvidia [23] implémente uniquemement les modèles de base BERT, GPT et SASrec [24] sans les optimiser pour les tâches de SBR [1]. Contrairement aux GNN qui ont connu des améliorations au fil des années pour le SBR les Transformers restent basiques. La problématique soulève donc la question de la pertinence des modèles Transfomer très peu exploités à ce jour.

### 1.1.4 Pré-entrainement et données non-ordonnées

L'utilisation de données non ordonnées pour le pré-entraînement des modèles Transformer pourrait également montrer des résultats prometteurs. Bien que ces données soient traditionnellement considérées comme inappropriées pour des tâches séquentielles à cause de l'absence d'ordonnancement, leur incorporation peut enrichir la compréhension globale du modèle. En pré-entraînant les modèles sur des données non ordonnées et en affinant ensuite avec des données séquentielles, il serait possible d'améliorer la robustesse et la performance des systèmes de recommandation.

1. Les résultats des Transformers de cette librairie appliqués à la SBR sur le jeu de données Yoochoose comparés à un GNN de base sont largement inférieurs.

### 1.1.5 Problématique

En poursuivant cette voie, notre étude vise à explorer et à évaluer les performances des modèles Transformer optimisés appliqués aux systèmes de recommandation SBR, avec une attention particulière à l'intégration de données non-ordonnées pour le pré-entraînement.

## 1.2 Objectifs de recherche

Les objectifs de cette recherche visent à améliorer les systèmes de recommandation basés sur les sessions en intégrant des techniques avancées de modélisation des séquences issues des modèles de NLP, en particulier les architectures Transformer. Les objectifs sont les suivants :

- Développer et évaluer des modèles Transformer adaptés à la SBR.
- Optimiser l'entraînement et les prédictions des Transformers.
- Intégrer les techniques d'optimisation des LLM récents dans nos modèles Transformer.
- Explorer les techniques de pré-entraînement pour améliorer les Transformers appliqués à la SBR.

## 1.3 Plan du mémoire

Ce mémoire est structuré en quatre chapitres, chaque chapitre traitant d'aspects spécifiques de la problématique de la prédiction du prochain article à l'aide des modèles Transformer.

Le premier chapitre présente une revue de la littérature sur les travaux antérieurs liés à la SBR et aux modèles de langage ainsi que une introduction aux modèles Transformer et au concept d'attention.

Le deuxième chapitre, nous introduisons les définitions et concepts de base liés à la SBR. Nous discutons des limitations des méthodes traditionnelles de masquage, des défis des modèles basés sur les sessions, de la prédiction du prochain article ainsi que de la comparaison avec les meilleurs modèles actuels qui sont les GNN.

Le troisième chapitre introduit le pré-entrainement des Transformers adapté à la SBR avec des données provenant d'autres distrubtion pour enrichir la compréhension de nos modèles de langage.

Le troisième chapitre introduit le pré-entrainement des Transformers adapté à la SBR avec des données provenant d'autres distrubtions pour enrichir la compréhension de nos modèles de langage. Nous discutons de l'intérêt du pré-entrainement dans le contexte de la SBR, examinons les liens avec les LLM, et évaluons les performances sur le jeu de données Yoochoose

ainsi que celui de l'entreprise ALC.

# CHAPITRE 2 REVUE DE LITTÉRATURE

Ce chapitre révise les principales notions de base, en commençant par une définition des concepts fondamentaux.

## 2.1 Définitions et concepts de base

### 2.1.1 La recommandation basée sur les sessions

La SBR consiste à recommander des produits ou du contenu en se basant uniquement sur les interactions effectuées par l'utilisateur au cours d'une session unique, sans historique utilisateur préalablement enregistré. Ce type de recommandation est particulièrement utile dans les contextes où l'historique long terme n'est pas disponible ou pertinent. Du fait que des utilisateurs peuvent êtes anonymes, on considère toutes les sessions totalement indépendantes même celles qui proviennent d'un même utilisateur.

Les domaines d'application de la SBR incluent par exemple le commerce électronique, où elle personnalise les suggestions de produits en temps réel, et le *streaming* de musique et de films, où elle recommande des contenus basés sur les interactions récentes des utilisateurs. Elle est également utilisée dans les réseaux sociaux pour proposer des amis et des groupes pertinents, ainsi que dans les jeux en ligne pour adapter les recommandations de jeux et améliorer l'engagement des utilisateurs.

### 2.1.2 Concepts du NLP

Le NLP est un domaine de l'intelligence artificielle qui se concentre sur l'interaction entre les ordinateurs et le langage humain. Il vise à permettre aux machines de comprendre, interpréter et générer le langage naturel de manière utile et significative.

#### *Token*

Le terme *token* est fondamental dans le domaine du NLP et des modèles de langage. Un *token* représente l'association d'un mot à un nombre qui le défini dans un vocabulaire (dictionnaire de *tokens*) donné.

### *Tokenization*

La *tokenization* est le processus de transformation d'un texte brut en une série de *tokens*. Cette étape est cruciale car elle définit les unités de données que le modèle utilisera pour apprendre et faire des prédictions.

Dans notre étude tout au long de ce mémoire, ce principe de *tokenization* sera utilisé sur des séquences d'articles de commerce électrnique plutôt que sur des phrases de mots en texte brut.

### Vocabulaire

Le vocabulaire fait référence à l'ensemble des mots ou *tokens* uniques qu'un modèle peut reconnaître et traiter. Le vocabulaire est souvent construit à partir des données d'entraînement, où chaque mot ou token est associé à un identifiant unique. La taille du vocabulaire influence la complexité du modèle.

### *Embedding*

Les *tokens* sont souvent transformés en vecteurs continus appelés *embeddings* dans un espace continu à grand nombre de dimensions. On y réfère par le terme *embeddings*, ou "plongements" en français [1]. Ces *embeddings* capturent des informations contextuelles et sémantiques des *tokens*. Différentes techniques existent pour générer ces *embeddings* :

- **Word2Vec :** génère des *embeddings* en utilisant des contextes locaux autour des mots.
- **GloVe :** génère des *embeddings* basés sur des statistiques globales des corpus de texte.
- **BERT :** utilise un modèle de Transformer pour créer des *embeddings* contextuels bidirectionnels.

### *Position Encoding*

Dans les modèles de Transformer, les *tokens* sont additionnés à des *position encodings* qui ajoutent des informations sur la position des *tokens* dans la séquence. Cela permet au modèle de capturer l'ordre des *tokens*, crucial pour comprendre le sens du texte.

1. Nous utiliserons dans ce mémoire les termes anglais plus familiers *embeddings* et *tokens* (jetons).

#### *Masked Token*

Dans le contexte des MLM, certains *tokens* dans une séquence sont masqués pour que le modèle apprenne à prédire ces *tokens* en se basant sur le contexte environnant. Cela renforce la compréhension contextuelle du modèle et améliore ses capacités de prédiction.

#### *Padding Token*

Lors du traitement de séquences de longueurs différentes, des *padding tokens* sont ajoutés pour uniformiser la longueur des séquences dans un lot (*batch*) de données. Ces *tokens* de padding sont ignorés par le modèle pendant l'apprentissage et la prédiction.

### 2.1.3 Le Mécanisme d'Attention

Au coeur de l'architecture du Transformer se trouve le mécanisme d'attention, qui est essentiel pour capturer les relations entre les éléments d'une séquence. L'attention permet au modèle de se concentrer sur différentes parties de la séquence d'entrée lorsque qu'il produit une représentation d'un élément. Le mécanisme d'attention utilisé dans le Transformer est appelé *scaled dot-product attention*.

Ce mécanisme fonctionne en calculant une série de scores d'attention qui déterminent l'importance relative des autres éléments de la séquence pour un élément donné. Voici les étapes principales du calcul de l'attention :

1. **Calcul des Scores d'Attention** : Pour chaque paire d'éléments dans la séquence d'entrée, un score est calculé en prenant le produit scalaire de leurs représentations, qui sont appelées les *keys* (K) et les *queries* (Q). Les scores sont ensuite divisés par la racine carrée de la dimension de la représentation pour stabiliser les gradients (*scaling*).
2. **Application de la Fonction Softmax** : Les scores d'attention sont transformés en probabilités à l'aide de la fonction softmax, ce qui normalise les scores et les convertit en poids d'attention. Chaque poids indique l'importance relative d'un élément de la séquence pour un autre élément.
3. **Pondération et Somme** : Les poids d'attention sont utilisés pour pondérer les valeurs (V) associées à chaque *key*, et les résultats pondérés sont ensuite sommés pour produire une représentation d'attention finale pour chaque élément de la séquence.

Le modèle Transformer utilise une variante appelée "attention multi-têtes", où plusieurs ensembles de *keys*, de *queries* et de valeurs sont utilisés en parallèle. Chaque ensemble, ou tête d'attention, apprend à se concentrer sur différentes parties de la séquence, permettant ainsi

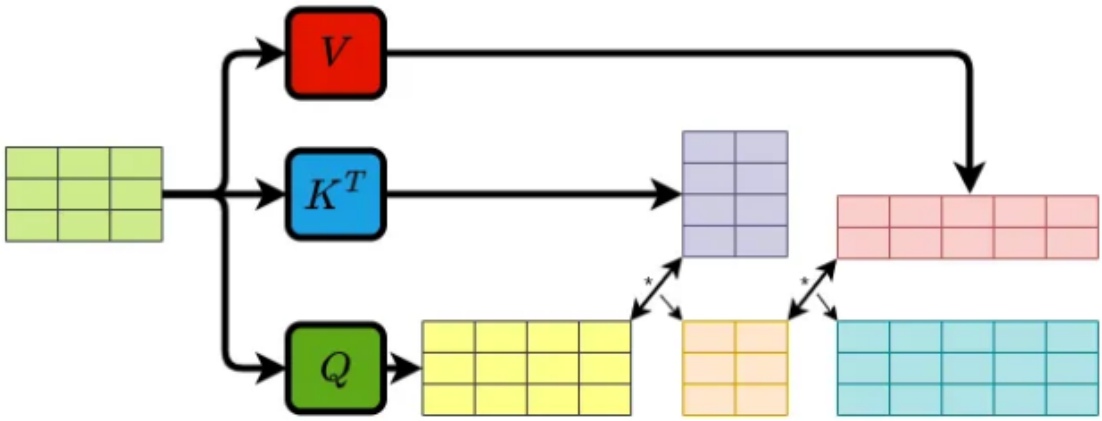


FIGURE 2.1 Représentation du mécanisme d'attention [1]

au modèle de capturer une plus grande variété de relations contextuelles. Les sorties de toutes les têtes d'attention sont ensuite concaténées et passées à une couche linéaire pour former la sortie finale du mécanisme d'attention. Ce mécanisme d'attention est crucial pour le succès du Transformeur, car il permet au modèle de traiter efficacement des séquences de longueur variable et de capturer des dépendances à longue portée sans les limitations des architectures séquentielles traditionnelles.

### 2.1.4 Le modèle Transformer

Les Transformers ont révolutionné le domaine du NLP grâce à leur capacité à modéliser des dépendances à longue portée dans les données textuelles. Introduits par Vaswani et al. en 2017 [2], les Transformers utilisent un mécanisme d'attention pour traiter simultanément toutes les parties d'une séquence, offrant une alternative plus efficace que les RNN et les réseaux réseaux de neurones convolutifs (CNN) pour de nombreuses tâches de NLP comme la prédiction du prochain mot, la prédiction du type de mot et la génération de texte.

#### Architecture du Transformer

L'architecture du Transformer se compose de deux parties principales : l'encodeur et le décodeur. Chaque partie est constituée de plusieurs couches identiques empilées, chacune contenant deux sous-couches : une couche d'attention multi-tête et une couche de réseau de neurones *fully connected*.

**Encodeur** L'encodeur du Transformer transforme une séquence d'entrée en une séquence d'*embeddings*. Chaque couche d'encodeur comprend :

— **Mécanisme d'auto-attention multi-tête :** permet au modèle de se concentrer sur différentes parties de la séquence d'entrée. L'attention de l'encodeur est souvent bidirectionnelle (il prend en compte le *token* précédent et suivant celui que l'on cherche à prédire).

— **Réseau de neurones feed-forward :** série de transformations non-linéaires qui affine les représentations. Il est utilisé après la couche d'attention pour s'ajuster sur les scores d'attention associés à chaque plongement/représentation.

**Décodeur** Le décodeur du Transformer génère une séquence de sortie à partir des représentations continues produites par l'encodeur. Les modèles basés sur des décodeurs sont souvent utilisés pour des tâches de génération de text comme GPT [18]. Chaque couche de décodeur comprend :

— **Mécanisme d'attention multi-tête :** permet de se concentrer à la fois sur la séquence d'entrée et sur la partie générée de la séquence de sortie. Le mécanisme d'attention du décodeur est souvent unidirectionnel (prend en compte uniquement les *tokens* qui précèdent celui que l'on cherche à prédire).

— **Réseau de neurones feed-forward :** affine les représentations avant de produire la sortie finale.

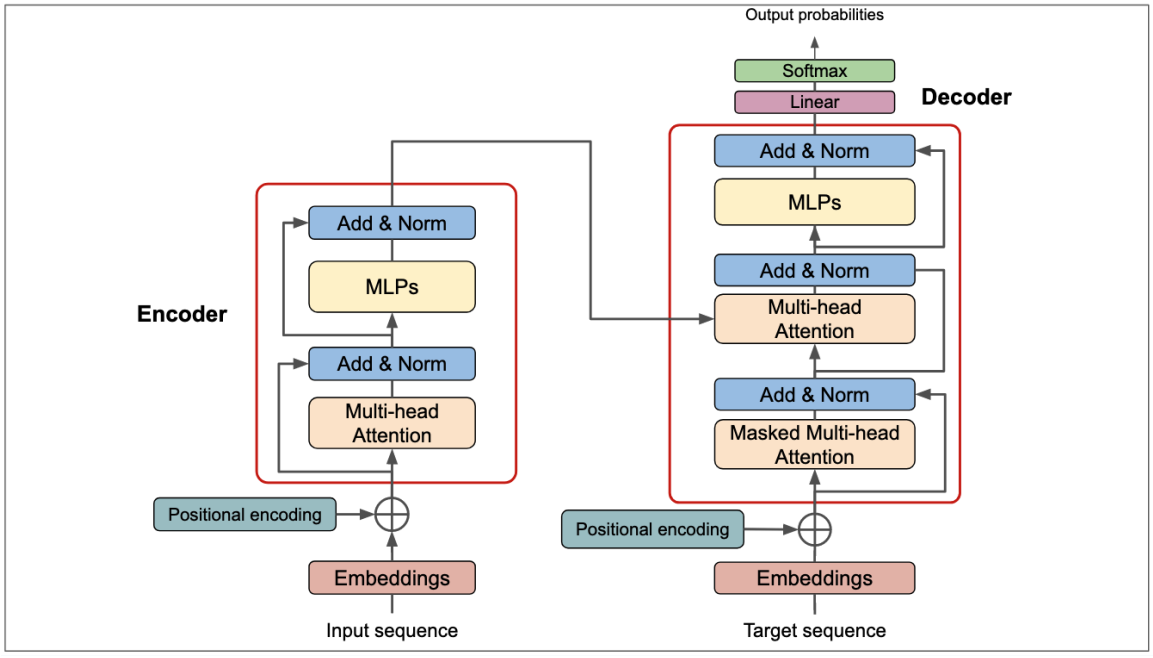


FIGURE 2.2 Architecture d'un Transformer de base [2]

Le Transformer est ainsi devenu la base de nombreux modèles de langage en NLP, y compris BERT, GPT, et leurs diverses itérations. Contrairement aux modèles séquentiels traditionnels tels que les RNN et les modèles à mémoire à long court terme (LSTM), le transformeur repose entièrement sur des mécanismes d'attention, pour modéliser les dépendances à long terme dans les séquences de données, en particulier l'attention multi-têtes qui le fait de manière plus efficace et flexible.

### 2.1.5 Les fonctions d'activation et les Transformers

Les fonctions d'activation jouent un rôle crucial dans les capacités d'ajustement de courbe des réseaux neuronaux. Elles introduisent des non-linéarités qui permettent aux modèles d'apprendre et de représenter des relations complexes entre les données d'entrée et de sortie. Elles influencent la convergence de l'entraînement, la propagation des gradients et la capacité des modèles à généraliser à partir des données d'entraînement. Dans le contexte des modèles de langage Transformer, les fonctions d'activation influencent directement les performances et l'efficacité des modèles. Nous discutons ici des fonctions d'activation couramment utilisées dans les LLM.

#### ReLU (Rectified Linear Unit)

La fonction ReLU (Rectified Linear Unit) [25] est définie comme suit :

$$\mathrm{ReLU}(x) = \max(0, x)$$

ReLU est largement utilisée en raison de sa simplicité et de son efficacité. Elle active uniquement les neurones dont l'entrée est positive, ce qui permet d'éviter la saturation des gradients et d'accélérer la convergence pendant l'entraînement. Cependant, ReLU peut souffrir du problème des neurones morts, où certains neurones ont toujours une valeur nulle en sortie, peu importe la valeur d'entrée.

#### GeLU (Gaussian Error Linear Unit)

La fonction GeLU (Gaussian Error Linear Unit) [26] combine les propriétés de ReLU avec celles de dropout et de zoneout. La fonction GeLU est définie par :

$$\mathrm{GeLU}(x) = x \cdot \Phi(x)$$

où $\Phi(x)$ est la fonction de répartition cumulative d'une distribution normale. GeLU permet des activations moins fortes et offre des avantages en termes de convergence et de performance par rapport à ReLU.

#### GLU (Gated Linear Unit) et variantes

Les Gated Linear Units (GLU) [27] et leurs variantes ajoutent une composante de *gating* qui permet de mieux contrôler le flux d'informations à travers les couches du réseau. La fonction GLU est définie comme suit :

$$\mathrm{GLU}(x, W, V, b, c) = (xW + b) \otimes \sigma(xV + c)$$

où $x$ est l'entrée de la couche, $W$ et $V$ sont des matrices de poids, $b$ et $c$ sont des biais, et $\sigma$ est la fonction sigmoïde. L'élément de *gating* $\sigma(xV + c)$ permet de moduler les activations de manière plus flexible.

D'autres variantes de GLU utilisées dans les LLMs incluent :

— **ReGLU (Rectified GLU)** :

$$\mathrm{ReGLU}(x, W, V, b, c) = \max(0, xW + b) \otimes \sigma(xV + c)$$

— **GEGLU (Gated Exponential Linear Unit)** :

$$\mathrm{GEGLU}(x, W, V, b, c) = (xW + b) \otimes \mathrm{GELU}(xV + c)$$

— **SwiGLU (Swish GLU)** :

$$\mathrm{SwiGLU}(x, W, V, b, c, \beta) = \mathrm{Swish}_\beta(xW + b) \otimes \sigma(xV + c)$$

où $\mathrm{Swish}_\beta(z) = z \cdot \sigma(\beta z)$ et $\beta$ est un paramètre de lissage.

### 2.1.6 Normalisation des couches dans les Transformers

La normalisation des couches conduit à une convergence plus rapide et est une composante intégrée des Transformers [28]. Plusieurs techniques de normalisation sont couramment utilisées dans les LLM, chacune ayant ses avantages et spécificités.

#### Normalization des couches

La **normalisation des couches (LayerNorm)** [28] est une technique où les activations d'une couche sont normalisées en fonction de leur moyenne et de leur écart-type. Cette méthode permet de stabiliser et d'accélérer l'entraînement en réduisant la variance de l'activation, ce qui aide le modèle à converger plus rapidement.

#### Normalisation RMS

Une autre méthode couramment utilisée est la **normalisation RMS (RMSNorm)** [29]. RMSNorm est similaire à LayerNorm, mais utilise uniquement la racine carrée de la moyenne des carrés des sortie des fonctions d'activation (RMS) pour normaliser les sorties de ces dernières. Cette technique offre des avantages en termes de stabilité et de performance lors de l'entraînement.

### 2.1.7 RNN

Les RNN sont un type de réseau de neurones conçu pour traiter des séquences de données en utilisant des boucles internes permettant de conserver des informations sur les états précédents. Ils sont particulièrement efficaces pour les tâches de traitement du langage naturel et les séries temporelles, car ils peuvent gérer des dépendances à long terme dans les données séquentielles.

### 2.1.8 GNN

Les GNN sont une classe de modèles de machine learning conçus pour traiter des données structurées sous forme de graphes, où les nœuds représentent des entités et les arêtes représentent les relations entre ces entités. Les GNNs propagent l'information à travers le graphe en agrégeant les caractéristiques des nœuds et de leurs voisins, leur permettant de capturer des informations contextuelles et relationnelles complexes.

### 2.1.9 LLM

Les LLM sont des architectures de traitement du langage naturel qui possèdent un très grand nombre de paramètres, souvent de l'ordre de plusieurs milliards. Ils sont entraînés sur d'énormes corpus de données textuelles, ce qui leur permet de capturer des nuances linguistiques complexes et de comprendre le contexte des mots dans une séquence. Grâce à leur taille et à la diversité des données d'entraînement, les LLM peuvent générer des réponses

cohérentes et pertinentes, offrant ainsi une polyvalence exceptionnelle pour diverses tâches de NLP, telles que la traduction automatique, la génération de texte et la réponse à des questions.

### 2.1.10 Prédiction du prochain *token*

Les LLM ont pour objectif principal de prédire le mot suivant dans une séquence de texte. Cette tâche est essentielle pour l'entraînement de ces modèles car elle leur permet de comprendre et de générer du texte de manière cohérente. La prédiction du prochain mot se base sur l'utilisation de mécanismes d'attention, où chaque mot de la séquence est pris en compte pour estimer le mot suivant le plus probable.

De la même façon, la prédiction du prochain article d'un utilisateur dans une session de navigation (SBR) repose sur l'analyse des interactions passées de l'utilisateur pour prédire son prochain choix. Bien que les données et le contexte diffèrent, le principe sous-jacent de prédiction reste comparable. Dès lors, on observe une similitude entre la prédiction du prochain mot en NLP et la prédiction du prochain article pour la SBR, le principe étant le même mais les données et leur traitement étant différents.


FIGURE 2.3 Prédiction du dernier mot d'une phrase [3]

## 2.2 L'évolution de la SBR

Les premières investigations sur la recommandation basée sur les sessions (SBR) se sont principalement concentrées sur l'examen des modèles séquentiels. Le modèle de chaîne de Markov [14], étant la forme la plus ancienne de recommandation de session, visait à prédire le prochain article en se basant sur un ou plusieurs articles précédents. Toutefois, cette méthode a tendance à négliger les informations relatives aux comportements de clics distants puisqu'elle prend principalement en compte la relation unidirectionnelle de la séquence de clics de l'utilisateur. En résumé, plus une séquence est longue moins les chaînes de Markov sont efficaces.

L'avènement de l'apprentissage profond a eu un impact positif dans le domaine de la SBR. Avec ce développement, les RNN sont apparus comme les méthodes de prédilection, avec GRU4REC [9] comme technique pionnière de RNN contribuant aux avancées de la SBR en 2015. Les RNN ont continué d'être développé en 2016 avec [30] qui propose l'intégration du temps de consultation des articles (*dwell time*) dans les recommandations basées sur les sessions, améliorant ainsi la précision des modèles de RNN en capturant mieux l'intérêt et l'engagement des utilisateurs.

Dans les développements récents des techniques de SBR, les modèles basés sur les graphes ont gagné en notoriété, atteignant régulièrement des performances supérieures. Cette évolution a commencé avec l'introduction de SC-GCN [13] en 2019 qui introduit la notion de graphe dans l'univers de la SBR. Puis ce concept a progressé au fil des années, illustrées par le modèle de graphe CARES [15], qui établit actuellement la référence en termes de scores de performance en passant par STAR [31], COTREC [17] et GCE-GNN [32]. Les GNN sont donc au sommet de la performance quand il s'agit de SBR et il y a de fortes chances qu'ils soient encore optimisés comme ils l'ont grandement été ces dernières années.

Dans le domaine des Transformer, des réalisations notables ont également été documentées dans le domaine de la SBR. Plus précisément, lors du RecSys Challenge 2022, une équipe utilisant des architectures Transformer pour le défi SBR a obtenu la 2e place parmi 300 équipes concurrentes [33] évalué sur un jeu de données propre à ce défi. De plus, une approche innovante a été introduite dans [34], où un modèle GNN intégrant un Transformer avec un module d'attention pour exécuter des séquences sous forme de graphe a été proposée. Ce Transformer intégré au GNN est basé sur une architecture de modèle BERT, qui est principalement basée sur un encodeur. Cette approche marque une fusion significative des modèles basés sur les graphes et des architectures de Transformer dans le domaine de la SBR. Finalement, l'entreprise *Meta* en collaboration avec *Nvidia* a développé une librairie en 2021 qui met à disposition des modèles Transformer avec des architecutres comme BERT au service de la SBR [23]. Dans la publication qui présente la librairie, ces derniers obtiennent des résultats de performances nettement moins bons que les GNN d'après leur évaluation sur le jeu de données Yoochoose [35]. Ces résultats sont également inférieurs à nos implémentations de BERT, GPT-2 et DeBERTa qui utilise une nouvelle technique de masquage qu'on détaillera dans les sections à venir. Cette librairie reste tout de même utile pour la mise à disposition d'une *pipeline* qui effectue le pré-traitement de données de session et l'entrainement de modèles de langage Transformer tout en un pour des personnes ne disposant pas d'implémentation Transformer.

## 2.3 Les architectures Tranformeur populaires

### 2.3.1 BERT

BERT (*Bidirectional Encoder Representations from Transformers*) [36] est un modèle Transformer basé sur l'architecture d'encodeur. La principale caractéristique de BERT est son utilisation d'une attention bidirectionnelle, ce qui signifie qu'il considère le contexte de tous les mots d'une phrase à la fois, aussi bien ceux qui précèdent que ceux qui suivent un mot donné. Cela permet de mieux comprendre le sens et les relations entre les mots dans une séquence.

La formule clé de l'attention utilisée dans BERT est l'attention *scaled dot-product*. Le calcul de cette attention est donné par :

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

où $Q$ est la matrice des *queries*, $K$ est la matrice des *keys*, $V$ est la matrice des valeurs, et $d_k$ est la dimension des *keys*. Cette attention est ensuite appliquée de manière multi-tête pour capturer diverses relations contextuelles :

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O \tag{2.2}$$

où chaque tête d'attention est calculée comme suit :

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.3}$$

### 2.3.2 GPT

GPT (*Generative Pre-trained Transformer*) [18] est basé sur une architecture de Transformer décodeur 2.4. Contrairement à BERT, GPT utilise une attention causale pour garantir que chaque mot ne peut se baser que sur les mots précédents dans la séquence. Cela rend GPT particulièrement adapté à la génération de texte.

La formule d'attention causale est similaire à celle de l'attention *scaled dot product* mais celle-ci est unidirectionnelle et ne regarde que les *tokens* qui précédent celui que l'ont veut prédire. On applique donc une modification pour s'assurer que les positions futures ne sont pas considérées :

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V \tag{2.4}$$

où $M$ est une matrice masque qui impose une attention causale.

GPT utilise également l'attention multi-tête pour capturer une variété de dépendances contextuelles :

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O \tag{2.5}$$

où chaque tête d'attention causale est calculée comme suit :

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.6}$$

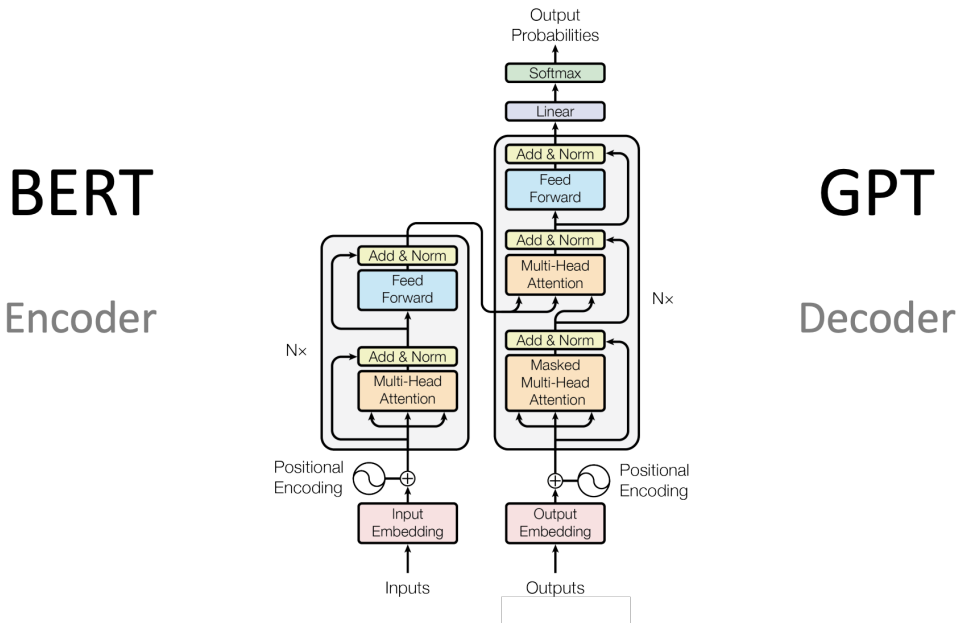


FIGURE 2.4 Architectures GPT et BERT [2]

### 2.3.3 DeBERTa

DeBERTa (*Decoding-enhanced BERT with disentangled attention*) [19] améliore BERT en introduisant l'attention désentrelacée et une meilleure représentation positionnelle. Dans DeBERTa, les mots sont représentés par des *embeddings* de contenu et des *embeddings* de position séparés.

La formule pour l'attention désentrelacée est :

$$\text{Attention}(Q, K, V, R) = \text{softmax}\left(\frac{QK^T + QR^T}{\sqrt{d_k}}\right)V \tag{2.7}$$

où $R$ représente les *embeddings* de position désentrelacés. Et comme l'attention régulière Q est la matrice des *queries*, K est la matrice des *keys*, V est la matrice des valeurs, et dk est la dimension des *keys*. Cette formule combine les informations de contenu et de position pour produire une attention plus précise.

DeBERTa utilise également l'attention multi-tête pour capturer diverses relations contextuelles

$$\text{MultiHead}(Q, K, V, R) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O \tag{2.8}$$

où chaque tête d'attention désentrelacée est calculée comme suit :

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V, RW_i^R) \tag{2.9}$$

Ces améliorations permettent à DeBERTa de capturer plus efficacement les relations sémantiques complexes dans les séquences de texte, surpassant ainsi les performances de BERT sur plusieurs tâches de NLP [4].

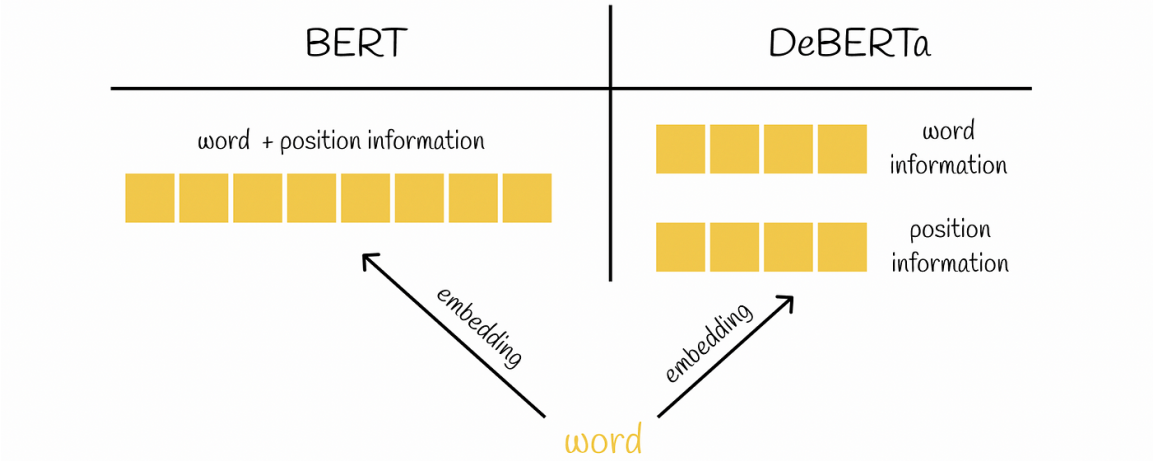


FIGURE 2.5 Différence entre BERT et DeBERTa [4]

## 2.4 Les techniques de masquage des modèles de langage

Le masquage est une technique essentielle dans les modèles de langage naturel qui consiste à cacher certaines parties d'une séquence d'entrée pour que le modèle apprenne à les prédire. Cette approche permet aux modèles de mieux comprendre le contexte et les dépendances entre les *tokens*, améliorant ainsi leur capacité à générer et à interpréter du texte. Le principe de masquage est particulièrement pertinent pour nos travaux, car il constitue une contribution spécifique dans notre recherche.

Dans cette section, nous décrivons les trois techniques de masquage utilisées dans les modèles de langage les plus populaires et efficaces : le MLM, le CLM, et le *permutative language modeling* (PLM). Chaque technique présente des avantages spécifiques pour l'apprentissage et l'application des modèles de langage naturels.

### 2.4.1 MLM

Le *Masked Language Modeling* MLM est une technique où certains tokens dans une séquence sont masqués aléatoirement, et le modèle doit prédire ces tokens masqués en utilisant le contexte environnant. Cette méthode permet au modèle d'apprendre des représentations contextuelles bidirectionnelles.

$$\mathcal{L}_{\text{MLM}} = -\sum_{i \in \mathcal{M}} \log P(x_i | x_{\setminus i})$$

où $\mathcal{M}$ est l'ensemble des positions masquées, $x_i$ est le token à la position $i$, et $x_{\setminus i}$ représente tous les tokens sauf $x_i$.

### 2.4.2 CLM

Le *Causal Language Modeling* (CLM), également connu sous le nom de modèle autoregressif, prédit chaque token dans une séquence en utilisant uniquement le contexte précédent. Comme chaque prédiction est basée sur les tokens qui précèdent le token cible, le modèle se limite à une attention unidirectionnelle.

$$\mathcal{L}_{\text{CLM}} = -\sum_{i=1}^{n} \log P(x_i | x_1, x_2, \ldots, x_{i-1})$$

Cette approche est utilisée pour des tâches de génération de texte, où chaque mot est généré séquentiellement en fonction des mots précédents.

### 2.4.3 PLM

Le *permutative language modeling* (PLM) est une technique utilisée par des modèles comme XLNet (variante de BERT). Ici, l'ordre des tokens est aléatoirement permuté, et le modèle doit prédire les tokens dans cet ordre permuté. Cela permet au modèle de capturer des dépendances bidirectionnelles tout en évitant certains des problèmes associés aux masquages classiques.

$$\mathcal{L}_{\text{PLM}} = -\sum_{i=1}^{n} \log P(x_{\pi(i)} | x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(i-1)})$$

où $\pi$ est une permutation aléatoire des indices des tokens dans la séquence.

Chaque technique de masquage présente des avantages distincts pour l'apprentissage des modèles de langage naturel. Le MLM permet une compréhension bidirectionnelle, le CLM est adapté pour la génération séquentielle, et le PLM combine les avantages des deux en utilisant des permutations aléatoires. Cette approche est utilisée pour des tâches de classification de texte, analyse de sentiments et recherche d'information entre autres [2].

## 2.5 L'état de l'art des LLM

Les LLM continuent d'évoluer rapidement, intégrant des techniques de pointe pour améliorer leurs performances et capacités. Les améliorations récentes se concentrent sur plusieurs aspects clés, notamment l'efficacité des architectures, les mécanismes d'attention avancés, et les stratégies de pré-entraînement et de fine-tuning. Parmi les techniques les plus innovantes, on trouve l'utilisation du *rotary positional encoding* (RoPE), les MoE et la pré normalisation des couches. Ces approches visent à maximiser l'efficacité de calcul tout en augmentant la qualité et la cohérence des prédictions des modèles.

Ces techniques et d'autres innovations sont intégrées dans les modèles de langage les plus récents, comme Mistral 7B, Gemma 7B, et Llama 3, qui représentent l'état de l'art en matière de traitement du langage naturel.

### 2.5.1 Mistral 7B

Mistral 7B [37] est un modèle de langage basé sur l'architecture transformeur, connu pour son efficacité et sa capacité à gérer de grandes séquences de texte. Mistral 7B utilise une variante

2. Pour la suite de cette étude l'approche de masquage PLM ne sera pas citée dans la partie évaluation car il ne se prête pas à la tâche de prédiction du prochain article.

avancée de l'attention multi-tête et intègre des techniques de compression des séquences pour réduire la complexité de calcul. Le modèle est conçu pour offrir des performances élevées sur une variété de tâches NLP, allant de la génération de texte à la traduction automatique. Mistral 7B se distingue par ses mécanismes d'attention optimisés et ses techniques avancées de régularisation, assurant une meilleure généralisation des modèles.

### 2.5.2 Gemma 7B

Gemma 7B [38] est un autre modèle de langage de pointe, optimisé pour la précision et l'efficacité. Il incorpore des améliorations en matière de mécanismes d'attention, avec un focus particulier sur l'encodage RoPE. Cette technique permet à Gemma 7B de capturer de manière plus efficace les relations contextuelles entre les tokens, en particulier pour les séquences longues. Gemma 7B utilise également des couches de normalisation améliorées et des techniques de régularisation avancées pour éviter le surapprentissage et améliorer la généralisation.

### 2.5.3 Llama 3

Llama 3 [39], introduit par Meta, est considéré comme le modèle de langage ouvert le plus performant à ce jour. Il combine plusieurs techniques de pointe pour améliorer à la fois la précision et l'efficacité. Llama 3 intègre l'encodage RoPE et des MoE, permettant au modèle de gérer une large gamme de tâches NLP avec une grande efficacité. De plus, Llama 3 utilise des optimisations avancées des Transformeurs, telles que des mécanismes d'attention améliorés et des techniques de régularisation sophistiquées. Cela permet une utilisation optimale des ressources de calcul tout en maintenant des performances élevées. Llama 3 a démontré des résultats impressionnants sur divers benchmarks, surpassant de nombreux modèles concurrents en termes de précision et de cohérence des prédictions.

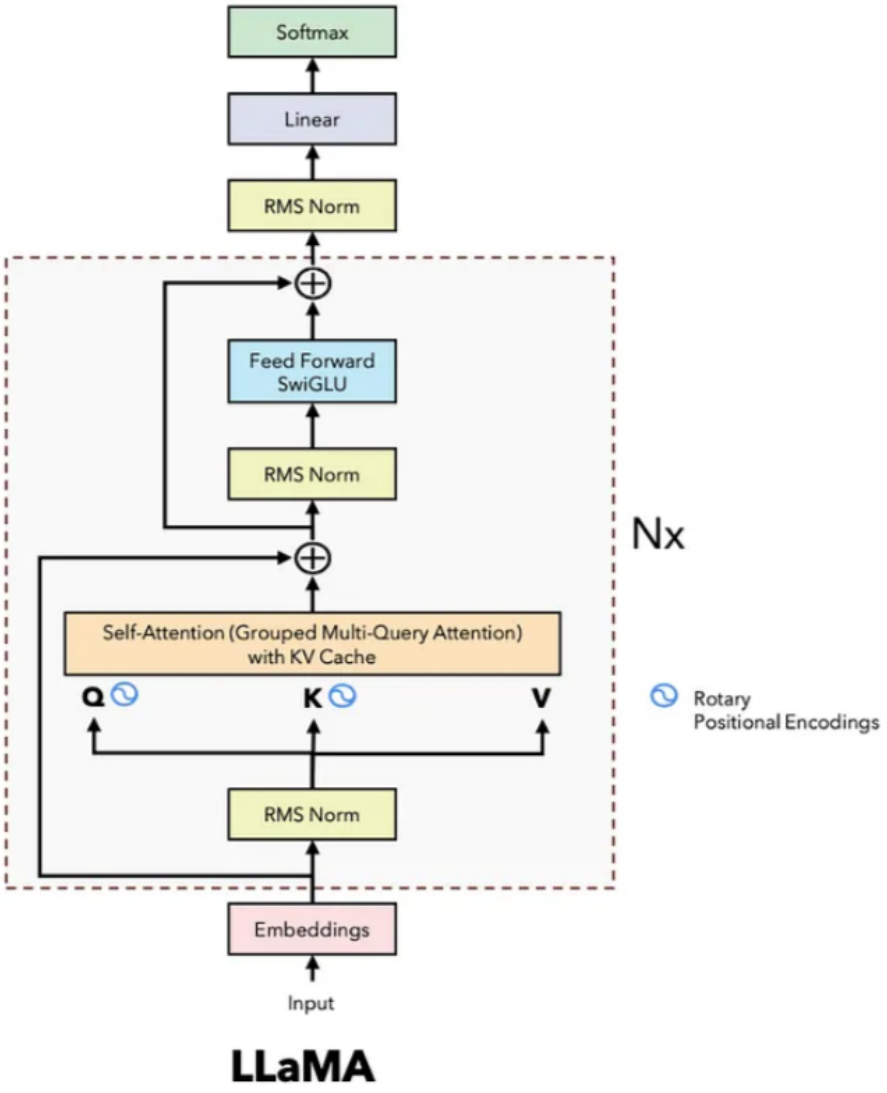


FIGURE 2.6 Architecture de Llama [5]

# CHAPITRE 3 OPTIMISATION DES TRANSFORMER POUR LA SBR

Les modèles de langage les plus récents [40] sont majoritairement des modèles pré-entrainés avec une architecture propre à chacun qui ont un point commun : les Transformers. En effet, ces derniers sont tous basés sur des types d'architectures de Transformer populaires. Parmi ces Transformer qui composent ces modèles de langage on retrouve les architectures suivantes : BERT [36], GPT [18] ainsi que DeBERTa [19] (une variante de BERT). Par exemple, Llama [39] possède une architecture composée de décodeurs (de plusieurs blocs GPT superposés) avec une technique d'attention causale (CLM) [41].

Ce chapitre porte sur la première contribution de notre travail et présente une technique d'optimisation majeure des modèles de langage basés sur des architectures Transformer connues, spécifiquement pour la tâche de prédiction du prochain article. Nous expliquerons le rôle et le contexte de ces techniques dans le cadre de la recommandation SBR.

## 3.1 Nouvelle approche de masquage adaptée à la SBR : SMM

Nous améliorons la SBR basée sur les Transformers en intégrant différentes techniques d'optimisation, en l'occurence une première technique d'optimisation basée sur le masquage qui est décrite dans cette section.

### 3.1.1 Prédiction du prochain article

La tâche de prédiction du prochain article, également appelée *next click prediction*, est cruciale pour les systèmes de recommandation basés sur les sessions. Cette tâche consiste à prédire l'article que l'utilisateur est le plus susceptible de cliquer ensuite, en se basant sur les interactions passées dans la session en cours.

Dans cette tâche, nous utilisons les modèles de langage du type Transformeurs pour capturer les relations séquentielles entre les articles cliqués dans une session. Les étapes de l'évaluation pour la prédiction du prochain article sont les suivantes :

1. **Préparation des données** : les données sont divisées en sessions d'entraînement et de test. Chaque session contient une séquence d'articles cliqués par l'utilisateur. Afin d'éviter de créer un biais du aux articles qui sont fréquemment sélectionnés en dernier, nous effectuons une opération d'augmentation de données qui consiste à générer des toutes les sous-séquences possibles à partir de toutes les séquences qui sont d'une longueur au moins égale à deux. Ces sous-séquences sont générées à partir de la séquence originale en retirant itérativement le

dernier article séléctionné par l'utilisateur. De cette manière tous les articles de la séquence sauf le premier seront masqués au moins une fois et pourront être prédits par notre modèle.

2. **Entraînement du modèle** : le modèle est entraîné sur les sessions d'entraînement pour apprendre les motifs de articles et les relations séquentielles entre les articles. Pour chaque séquence c'est toujours le dernier article cliqué qui est masqué et on calcule la perte cross-entropie uniquement sur cet article masqué.

3. **Prédiction** : pour chaque session de test, le modèle prédit l'article suivant basé sur les articles précédemment cliqués dans la même session. Les prédictions sont ensuite comparées aux articles consultés par l'utilisateur pour évaluer la performance du modèle en utilisant des métriques comme la Précision@20 et le Rang réciproque moyen (MRR). Par exemple, sur la figure 3.1 on recommande quatres articles (on aurait donc un score de Précision@4). Si le dernier article acheté par l'utilisateur est parmi ces recommandations (basées sur la session ne contenant pas le dernier article), alors on condière que la prédiction est correcte.

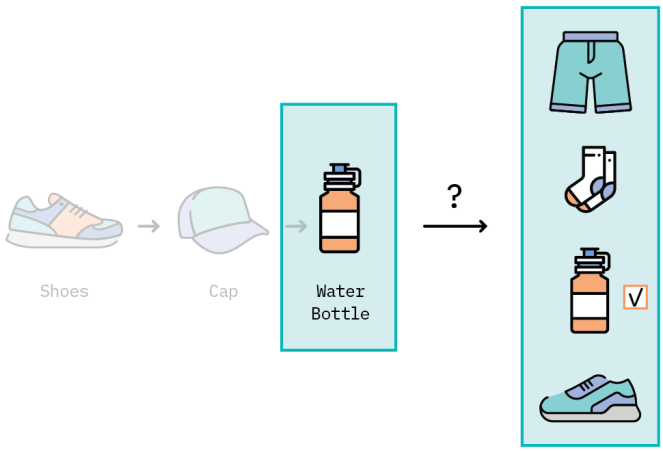


FIGURE 3.1 Tâche de prédiction du prochain article [6]

La prédiction du prochain article est une tâche complexe mais essentielle pour améliorer la pertinence des recommandations dans les systèmes basés sur les sessions. Les performances des modèles sur cette tâche sont souvent utilisées pour mesurer leur efficacité globale dans les systèmes de recommandation.

### 3.1.2 Description de la nouvelle approche de masquage

Dans cette section, nous introduisons une nouvelle approche de masquage (d'entraînement) pour la recommandation basée sur session appelée *Sequential Masked Modeling* (SMM). Cette

technique est conçue pour améliorer les performances de prédiction du prochain article avec des Transformers en utilisant une technique innovante de masquage lors de l'entrainement. La suite de cette section présente la théorie mais pour une meilleure compréhension de la méthode nous avons réalisé la section 3.1.3 pour l'illustrer avec un exemple plus concret.

L'approche SMM fonctionne en appliquant trois principes majeurs :

***Data augmentation***

Pour cette approche on utilise une *sliding window* (fenêtre coulissante) pour générer des sous-séquences à partir de séquences initiales en les découpant en morceaux de taille fixe, généralement déterminée par la longueur maximale acceptée par le modèle, notée *max_len*.

Concrètement, pour une séquence initiale $s = (x_1, x_2, \ldots, x_n)$ et une taille de fenêtre *max_len*, la fenêtre se déplace le long de la séquence pour générer des sous-séquences de la manière suivante :

- **Étape 1** : la première sous-séquence est prise à partir de la fin de la séquence initiale. Si la longueur de la séquence est inférieure à *max_len*, la sous-séquence contient tous les éléments. Par exemple, $s_1 = (x_{n-max\_len+1}, \ldots, x_n)$.
- **Étape 2** : la fenêtre glisse ensuite d'un pas d'un élément vers la gauche pour créer la sous-séquence suivante. Par exemple, $s_2 = (x_{n-max\_len}, \ldots, x_{n-1})$.
- **Étape 3** : ce processus continue jusqu'à ce que la fenêtre atteigne le début de la séquence initiale. Par exemple, $s_3 = (x_{n-max\_len-1}, \ldots, x_{n-2})$, et ainsi de suite.

En utilisant cette technique, chaque token de la séquence initiale $s$ est inclus dans plusieurs sous-séquences, permettant ainsi au modèle de voir chaque token dans différents contextes d'entraînement. On définit chaque sous-séquence comme ceci :

$$s_i = (x_{n-max\_len-i+2}, \ldots, x_{n-i+1}) \quad \text{pour} \quad i = 0, 1, 2, \ldots, n - max\_len$$

Cette technique est particulièrement utile pour la majorité des modèles de réseaux de neurones où la longueur des séquences d'entrée est limitée par *max_len*. En appliquant la *sliding window*, nous nous assurons que toutes les parties pertinentes de la séquence initiale sont utilisées pour l'entraînement, ce qui améliore la robustesse et la performance du modèle. Cette technique de génération de sous-séquences se nomme : *data augmentation* (augmentation de données).

### Masquer l'avant dernier *token*

Pour chaque séquence initiale $s = (x_1, x_2, \ldots, x_n)$, nous générons à l'aide d'une *sliding window* des sous-séquences $s_i = (x_1, x_2, \ldots, x_i)$ pour $i = 1, 2, \ldots, n-1$. Chaque sous-séquence ainsi générée a son **avant-dernier *token*** $x_{i-1}$ masqué, et la perte est calculée uniquement sur ce *token* masqué. Cette approche permet au modèle de bénéficier d'un minimum de contexte à droite grâce à l'attention bidirectionnelle propre aux architectures de type BERT, ce qui est bénéfique lors de l'apprentissage. Même si dans la réalité la prédiction du prochain article ne possède pas de contexte à droite (car on ne connait pas le prochain article par rapport à celui sur lequel on évalue), cette approche est bien adaptée à un contexte où l'ordre des items n'est pas fortement contraint et où c'est plutôt la proximité de paires d'items qui importe, comme c'est le cas pour des données de commerce électronique et contrairement à des données linguistiques.

De plus, afin de permettre à notre modèle de s'ajuster par rapport au dernier *token* de chaque séquence initiale $s = (x_1, x_2, \ldots, x_n)$, nous générons une séquence où le dernier *token* est un *token* [CLS] (un *token* qui signifie la fin de la séquence). Dès lors, en masquant l'avant-dernier *token* de chaque séquence générée, tous les tokens de la séquence initiale seront utilisés séquentiellement pour l'entrainement de notre modèle.

$$s_i = (x_1, x_2, \ldots, [\text{MASK}], x_i)$$

La fonction de perte utilisée dans SMM est définie comme suit :

$$\mathcal{L}_{\text{SMM}} = -\log P(x_{i-1} | x_1, x_2, \ldots, x_{i-2}, x_i)$$

Ainsi, grâce à la *data augmentation* et à notre approche de masquage, tous les tokens d'une séquence initiale $s$ se retrouvent masqués un par un, et le modèle peut s'ajuster selon les éléments qui le précèdent et un minimum de contexte à droite provenant de l'élement suivant lors de l'entrainement.

Dans les sous-sections à venir nous formulons des hypothèses qui seront ensuite vérifiées dans la partie 3.3.

### Avantage par rapport à l'approche CLM

Nous faisons l'hypothèse qu'un des principaux avantages de notre approche SMM par rapport au *Causal Language Modeling* (CLM) de l'architecture GPT est que le mécanisme d'attention

change de manière significative. Dans un modèle CLM, l'attention est unidirectionnelle, c'est-à-dire que chaque token ne peut "regarder" que les *tokens* qui le précèdent. Cela limite la capacité du modèle à capturer des relations complexes entre les *tokens*, car le contexte complet de la séquence n'est pas pris en compte.

$$\text{Attention(CLM)} = \sum_{j=1}^{i} \alpha_{ij} x_j, \quad \text{où } \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{i} \exp(e_{ik})}$$

où :

- $i$ est la position du token en cours de traitement,
- $x_j$ est la représentation du token à la position $j$,
- $\alpha_{ij}$ est le score d'attention entre les tokens aux positions $i$ et $j$,
- $e_{ij}$ est la compatibilité entre les tokens aux positions $i$ et $j$, calculée par un produit scalaire.

En revanche, avec SMM, chaque *token* dans la séquence peut regarder partout, grâce à la *data augmentation* qui génère des sous-séquences. Cela permet au modèle d'apprendre des représentations plus riches et plus contextuelles, car chaque token a accès à l'ensemble du contexte séquentiel et donc aura un score d'attention plus précis.

$$\text{Attention(SMM)} = \sum_{j=1}^{n} \alpha_{ij} x_j, \quad \text{où } \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})}$$

où :

- $n$ est la longueur totale de la séquence,
- Les autres variables sont définies de la même manière que précédemment.

Le mécanisme d'attention bidirectionnelle permet une meilleure capture des dépendances à long terme, ce qui est crucial pour les tâches de recommandation basée sur session où les interactions entre items peuvent être complexes et étendues. Cette approche améliore non seulement l'apprentissage des dépendances complexes entre les items, mais elle permet également au modèle de mieux généraliser lors de la prédiction du prochain article.

Puisque le masquage CLM est associé au mécanisme d'attention *Causal* qui ne prend jamais en compte les *tokens* futurs lors du calcul des scores d'attention de chaque *token*, notre approche ne s'applique pas à l'architecture GPT. C'est pourquoi, dans la partie résultats, nous utiliserons GPT pour nous comparer à l'approche CLM avec notre approche SMM implémentée sur les architectures BERT et DeBERTa.

**Avantage par rapport à l'approche MLM**

Pour justifier l'intérêt de notre approche SMM par rapport à l'approche MLM, il est crucial de considérer le paramètre *max_len* (taille maximale) de notre modèle.

En effet, chaque modèle de réseau de neurones multicouches possède une taille maximale fixe pour ses séquences de données en entrée. Il est inévitable que certaines séquences en entrée soient plus longues que la taille *max_len* de notre modèle. Comme démontré par Vaswani et al. [2], la complexité computationnelle des modèles de type Transformer augmente quadratiquement avec la taille de la séquence, ce qui rend les grandes séquences particulièrement coûteuses en termes de calcul. Bien que cet article ne traite pas explicitement de l'augmentation exponentielle du temps de calcul, il souligne la complexité quadratique de l'auto-attention en fonction de la longueur de la séquence, illustrant ainsi comment le temps de calcul peut devenir prohibitif pour de grandes séquences.

Supposons maintenant qu'une séquence soit de taille deux fois supérieure au paramètre *max_len* de notre modèle. Pour appliquer la technique de masquage MLM, il serait nécessaire de découper cette séquence en deux parties de taille *max_len*. Les deux *tokens* situés respectivement à la fin de la première séquence et au début de la seconde sont censés être consécutifs. Dès lors, la nécessité de diviser la séquence entraine une perte d'information (puisque le modèle ne verra plus ces deux *tokens* comme étant consécutifs) et, par conséquent, une baisse de performance. L'idée de découper la séquence trop longue en sous-séquences qui se chevauchent pourrait sembler être une solution. Cependant, étant donné que le masquage est aléatoire, cette méthode créerait un biais en faveur des *tokens* situés au milieu et au début de la séquence initiale, car ils apparaîtraient plus souvent dans les sous-séquences, contrairement aux derniers *tokens* qui ne seraient présents qu'une seule fois.

### 3.1.3 Illustration de SMM et MLM avec $max\_len = 5$

Ici les sous-séquences en couleur réprésentent les séquences de taille maximales pouvant être introduites dans notre modèle.

Avec [M] = *Mask*.

**SMM :**

Dans l'approche SMM, chaque séquence est dupliquée une fois en interchangeant le dernier et l'avant dernier élément puis réduite jusqu'aux deux derniers éléments restants avec une fenêtre coulissante (*sliding window*). Afin d'appliquer le masquage sur tous les tokens de la

séquence initiale on découpe une séquence de taille $max\_len = 5$ **à la fin** de chaque sous séquence générée. Prenons la première ligne représentant une séquence d'articles (consultés durant une session) et numérotés selon le vocabulaire de notre *tokenizer* personnalisé. Les séquences en bleu des lignes suivantes représentent les données augmentées utilisées pour l'entraînement :

$[5, 6, 8, 4, 98, 56, 32, 12, 78, 54, 74, 23, 56, [M], [CLS]]$
$[5, 6, 8, 4, 98, 56, 32, 12, 78, 54, 74, 23, [M], 57]$
$[5, 6, 8, 4, 98, 56, 32, 12, 78, 54, 74, [M], 56]$
$[5, 6, 8, 4, 98, 56, 32, 12, 78, 54, [M], 23]$
$[5, 6, 8, 4, 98, 56, 32, 12, 78, [M], 74]$
$[5, 6, 8, 4, 98, 56, 32, 12, [M], 54]$
$[5, 6, 8, 4, 98, 56, 32, [M], 78]$
$[5, 6, 8, 4, 98, 56, [M], 12]$
$[5, 6, 8, 4, 98, [M], 32]$
$[5, 6, 8, 4, [M], 56]$
$[5, 6, 8, [M], 98]$
$[5, 6, [M], 4]$
$[5, [M], 8]$
$[[M], 6]$

Avec cette approche on peut envoyer les entrées masquées séquentiellement au modèle afin que tous les tokens de la séquence initiale soient masqués un par un avec un **maximum** de contexte possible.

### MLM sans augmentation de données

Dans l'approche MLM sans augmentation de données, le masquage est aléatoire. Voici un exemple de séquence manipulée (avec $max\_len = 5$ on se retrouve avec deux sous-séquences masquées aléatoirement de taille 5) :

— $[5, [M], 8, 4, [M]], 56, 32, 12, 78, 54, [M], 23, 56, 98]$

On remarque que certains *tokens* comme le *token* 54 ne possède pas suffisement de contexte pour un calcul de score d'attention optimal contrairement à l'approche SMM ci-dessus.

### MLM avec *data augmentation* introduisant un biais

Nous pourrions alors nous poser la question : est ce que la *data augmentation* pourrait aider l'approche MLM comme elle a aidé pour SMM ?

Dans l'approche MLM avec *data augmentation* et *window sliding*, chaque séquence est réduite jusqu'aux deux derniers éléments restants mais le masquage est aléatoire :

$[5, 6, 8, 4, 98, 56, 32, 12, 78, 54, 74, 23, [M], 98]$
$[5, 6, 8, 4, 98, 56, 32, 12, [M], [M], 74, 23, 56]$
$[5, 6, 8, 4, 98, 56, 32, 12, 78, 54, [M], [M]]$
$[5, 6, 8, 4, 98, 56, 32, [M], 78, [M], 74]$
$[5, 6, 8, 4, 98, 56, 32, 12, 78, [M]]$
$[5, 6, 8, 4, 98, [M], 32, 12, [M]]$
$[5, 6, 8, 4, 98, [M], 32, 12]$
$[5, 6, [M], 4, [M], 56, [M]]$
$[5, 6, [M], 4, 98, 56]$
$[5, [M], 8, 4, [M]]$
$[[M], 6, 8, [M]]$
$[5, 6, [M]]$
$[[M], 6]$

On remarque ici que les derniers *tokens* de la séquence (comme le *token* 98) ont beaucoup moins de chances d'être masqués que ceux au milieu. Cette approche créé donc un biais en faveur de certains *tokens*. La prédiction de notre modèle sera donc moins précise lorsqu'il s'agira de prédire les derniers *tokens* d'une longue séquence.

De plus, puisque le masquage est aléatoire dans le cas MLM il se peut que des *tokens* qui sont au début de la séquence soient masqués. Or, cela ne sera pas bénéfique pour l'entrainement de notre modèle puisque celui-ci est évalué sur la prédiction du prochain article et non des articles prédécents on préfère donc masquer un article vers la fin de la séquence plutôt qu'au début comme le fait notre approche SMM.

Ces hypothèses seront vérifiées dans la section 3.3.

### SMM en résumé

L'approche SMM offre une approche efficace pour la prédiction du prochain article en utilisant un mécanisme de masquage de l'avant-dernier *token*, de *data augmentation* et de *window sliding*, permettant ainsi une amélioration significative des performances par rapport aux

modèles de masquage classiques.

## 3.2 Techniques de LLM adaptées à la SBR

En plus de l'amélioration du SBR par des techniques de masquage, nous avons aussi implémenté des techniques récentes de LLM que nous avons intégrées à notre architecture BERT. Cette section décrit ce second volet d'amélioration. Nous verrons enfin dans la section évaluation 3.4.3 comment chacune de ces techniques a aidé ou non à la performance de notre modèle.

### 3.2.1 *Weight Tying*

Le *Weight Tying*, ou partage de poids, est la première de quatre techniques intégrées à nos architectures BERT, GPT et DeBERTa. C'est une technique utilisée dans les modèles Transformer pour réduire le nombre de paramètres et améliorer la performance. Cette technique consiste à utiliser les mêmes poids pour les *embeddings* d'entrée et les poids de la couche de sortie du modèle. En d'autres termes, les matrices de poids utilisées pour encoder les tokens en *embeddings* sont les mêmes que celles utilisées pour prédire les tokens à partir de ces vecteurs de représentation dans la sortie de notre encodeur.

Soit $W_e$ la matrice d'*embeddings* de taille $V \times d$, où $V$ est la taille du vocabulaire et $d$ est la dimension des *embeddings*. Les *embeddings* d'entrée pour un *token* $t_i$ sont donnés par $W_e[t_i]$.

Dans le MLM, la tâche est de prédire le *token* masqué à partir de la représentation contextuelle $h_i$ générée par le modèle. Cela se fait généralement en utilisant une couche linéaire suivie d'une fonction softmax. La sortie de cette couche linéaire, avant l'application de la fonction softmax, est donnée par $W_o h_i + b$, où $W_o$ est la matrice de poids de sortie de taille $V \times d$ et $b$ est le biais.

Avec le *Weight Tying*, nous imposons $W_o = W_e^T$. Ainsi, la prédiction pour le *token* masqué est donnée par :

$$\hat{y}_i = \text{softmax}(W_e^T h_i + b)$$

L'utilisation de la même matrice de poids pour les *embeddings* et la couche de sortie permet non seulement de réduire le nombre de paramètres, mais aussi d'améliorer la généralisation du modèle en liant étroitement les *embeddings* d'entrée et de sortie.

Avantages du *Weight Tying* :

— Réduction du nombre de paramètres : En partageant les poids entre les embeddings et la couche de sortie, le modèle nécessite moins de paramètres, ce qui peut réduire le risque de surapprentissage et améliorer l'efficacité.
— Meilleure généralisation : En utilisant les mêmes poids pour les deux tâches, le modèle est encouragé à apprendre des représentations plus cohérentes et robustes.

### 3.2.2 Pré-normalisation des couches

La pré-normalisation des couches est la seconde technique implémentée sur toutes nos architectures de Transformers compatibles à savoir GPT, BERT et DeBERTa.

Les LLM utilisent la **pré-normalisation des couches (*pre-layer normalization*)** [7], qui applique la normalisation avant la couche d'attention multi-têtes et avant la couche *feed forward* 3.2. La pré-normalisation a démontrée une meilleure stabilité d'entraînement dans les LLM. En appliquant la normalisation avant l'opération d'attention, le modèle peut mieux gérer les gradients pendant l'entraînement, ce qui conduit à une convergence plus stable. Par exemple Llama 3 [39] incorpore cette technique dans son architecture.

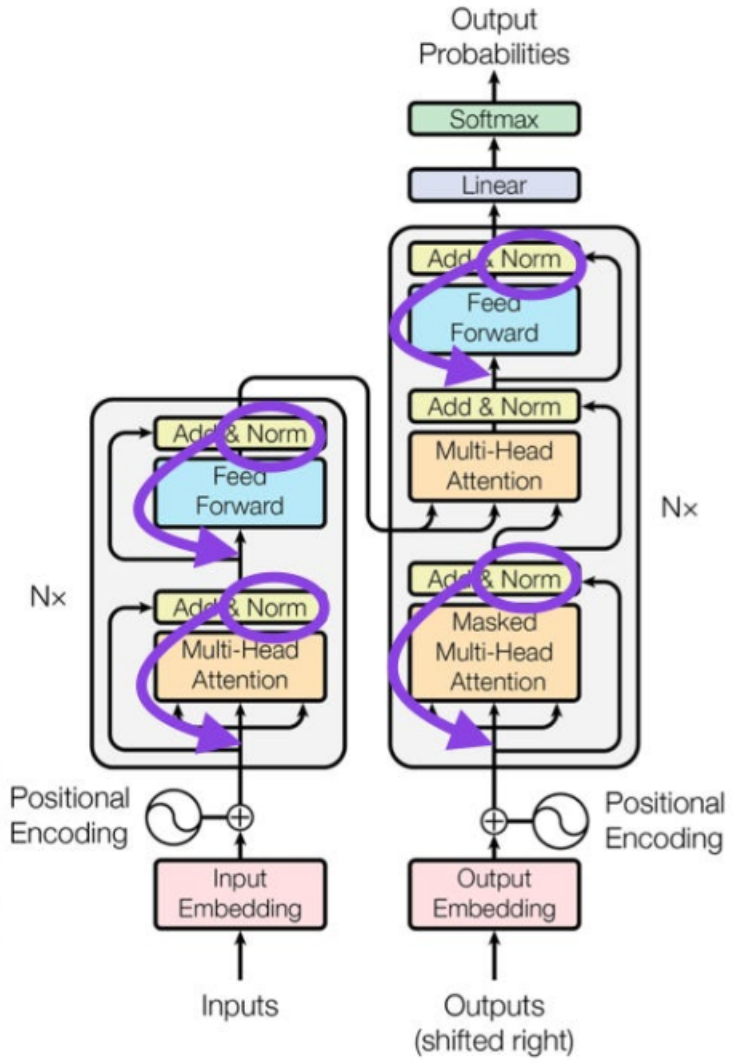


FIGURE 3.2 Pré normalisation des couches [7]

### 3.2.3 Mixture d'experts, MoE

Plusieurs modèles LLM utilisent l'architecture MoE, dont Llama 3 [39]. Cette architecture permet d'augmenter considérablement la capacité d'un modèle sans augmenter proportionnellement les coûts de calcul. C'est la troisième technique que nous avons implémenté sur nos architectures compatibles BERT et DeBERTa.

Dans cette approche, un "expert" est simplement un réseau *feed-forward*, et un routeur est chargé de diriger les *tokens* vers le bon expert. Pour ce faire, une couche linéaire calcule les probabilités d'assignation des états cachés à chaque expert, représentant ainsi la pertinence de chaque expert pour traiter ces *tokens*. Les experts sont ensuite sélectionnés en fonction de ces probabilités, et les états cachés résultants sont obtenus par la somme pondérée des sorties de ces experts.

Avantages de l'architecture MoE :

— Elle permet une meilleure utilisation des ressources en répartissant le traitement des tokens entre différents experts, ce qui réduit la charge computationnelle tout en augmentant la capacité du modèle.
— Cette approche améliore la capacité du modèle à capturer des relations complexes dans les données, ce qui se traduit par des performances accrues sur diverses tâches de traitement du langage naturel.

**Fonctionnement du routeur** Le routeur joue un rôle central dans l'architecture MoE. Il prend en entrée les états cachés des *tokens* et calcule les probabilités d'assignation à chaque expert, représentant ainsi la pertinence de chaque expert pour traiter ces *tokens*. La couche linéaire utilisée pour ce mapping est définie comme suit :

$$\text{Probabilité} = \text{softmax}(W \cdot \text{hidden\_state} + b)$$

où $W$ et $b$ sont des paramètres appris du routeur. Les experts sont ensuite sélectionnés en fonction de ces probabilités, ce qui permet d'optimiser la répartition des *tokens* à différents experts.

**Combinaison des sorties des experts** Une fois les experts sélectionnés, les sorties de ces experts sont combinées pour former l'état caché final. La combinaison se fait par une somme pondérée des sorties des experts sélectionnés, définie comme suit :

$$\text{Output} = \sum_{i=1}^{N} (\text{Probabilité}_i \cdot \text{Output}_{\text{expert}_i})$$

où $N$ est le nombre total d'experts, et $\text{Output}_{\text{expert}_i}$ est la sortie du $i$-ème expert.

L'architecture MoE est donc une avancée significative dans la conception des Transformeurs. Cette technique permet non seulement d'augmenter la capacité du modèle en ajoutant plus d'experts, mais aussi d'améliorer l'efficacité en ne faisant appel qu'à une fraction des experts à chaque étape de traitement, ce qui réduit les coûts de calcul.

### 3.2.4 Encodage Positionnel Contextuel

Cette quatrième technique que nous avons implémenté est une amélioration très récente dérivée de RoPE [42] (qui est utilisée dans la majorité des LLM les plus populaires). Dans l'Encodage Positionnel Contextuel (CoPE) [43], les positions sont mesurées de manière dépendante du contexte plutôt que par un simple décompte de *tokens*. Nous avons implémenté cette technique uniquement sur notre architecture BERT pour une raison de compatibilité. Cette méthode détermine d'abord quels *tokens* doivent être inclus lors de la mesure de distance en utilisant leurs vecteurs contextuels. Une *gate value* $g$ est calculée pour chaque paire de *query* Q et de *key* K (les mêmes Q et K qui ont été présentés dans la définition de l'attention) :

$$g(q, k) = \sigma(q \cdot W_g \cdot k^T)$$

où $\sigma$ est la fonction sigmoïde. Une *gate value* de 1 signifie que la *key* K sera comptée dans la mesure de position, tandis qu'une valeur de 0 signifie qu'elle sera ignorée. Cela permet au modèle d'inclure les *tokens* pertinents contextuellement lors du calcul des positions.

Ensuite, les valeurs de position sont calculées en ajoutant les *gate values* entre le *token* actuel et le *token* cible :

$$p_{i,j} = \sum_{k=i}^{j-1} g(q, k)$$

Si les *gates* sont toujours à 1, *contextual position encoding* (CoPE) retrouve les positions relatives originales d'une séquence de *tokens*, ce qui en fait une généralisation de l'*encodage de position relative*. Les valeurs de position peuvent être des comptages de types de mots spécifiques ou d'autres concepts utiles pendant l'entraînement.

Contrairement aux positions de *tokens* traditionnelles, les valeurs de position de CoPE peuvent prendre des valeurs fractionnaires en raison de la fonction sigmoïde. Au lieu d'utiliser une couche d'*embedding* pour convertir les valeurs de position en vecteurs, CoPE utilise une interpolation entre les valeurs entières. Chaque position entière se voit attribuer un vecteur d'*embedding* apprenable, et l'*embedding* pour la position $p$ est interpolé à partir des *embeddings* entiers les plus proches :

$$E(p) = (1 - \alpha)E(\lfloor p \rfloor) + \alpha E(\lceil p \rceil)$$

où $\alpha = p - \lfloor p \rfloor$ est la partie fractionnaire de $p$.

Enfin, les poids d'*attention* sont calculés de manière similaire aux méthodes standard :

$$A(q, k) = \text{softmax}\left(\frac{q \cdot (k + E(p_{i,j}))}{\sqrt{d_k}}\right)$$

Cette approche permet à notre modèle Transformer d'encoder efficacement les informations positionnelles, améliorant la capacité du modèle à capturer des relations complexes entre les *tokens*.

## 3.3 Évaluation et expérimentations

Les différentes techniques que nous avons développées font l'objet d'une série d'expériences afin d'évaluer leur efficacité. Nous décrivons ici les détails méthodologiques des expérimentations ainsi que les résultats des évaluations.

### 3.3.1 Ensembles de données des expérimentations

Pour évaluer et comparer les performances de ces différents modèles de langage Transformer avec notre nouvelle approche de masquage et sur les nouvelles techniques d'optmisation, nous avons choisi les trois ensembles de données Yoochoose [35], Tmall [44] et Diginetica [45] qui sont des jeux de données populaires dans la recommandation basée sur les sessions.

#### Pré-traitement des données

Les ensembles de données font l'objet d'un pré-traitement afin de rendre les résultats comparables à différentes études du SBR, notamment celles qui peuvent être considérées comme l'état de l'art et utilisent les GNN [13] et [15].

Nous utilisons trois ensembles de données disponibles publiquement pour montrer l'efficacité de notre méthode. Le pré-traitement se déroule en trois étapes principales pour correspondre aux statistiques du tableau 4.1 :

— Nous ordonnons toutes les sessions par ordre chronologique et divisons les données en données d'entraînement et en données de test en fonction des horodatages des sessions.
— Nous filtrons les items apparaissant moins de 5 fois ou apparaissant uniquement dans l'ensemble de test, ainsi que les sessions de longueur un.
— Nous effectuons une augmentation des données par avec une *sliding window* de taille 10 pour générer davantage d'échantillons de données dans une session. Pour une session [v1, v2, ..., vn], nous générons les échantillons ([v1, v2, ..., vn-1], vn), ..., ([v1, v2], v3), ([v1], v2).

Tableau 3.1 Statistiques des jeux de données

| Dataset | Diginetica | Tmall | Yoochoose1 64 |
|---|---|---|---|
| #Train sessions | 719 470 | 351 268 | 369 859 |
| #Test sessions | 60 858 | 25 898 | 55 898 |
| #Articles | 43 097 | 40 728 | 16 766 |
| *Avg. lengths* | 5.12 | 6.69 | 6.16 |

### 3.3.2 Implémentation

Nous avons implémenté, à l'aide de la librairie *PyTorch*, tous les modèles de langage Transformeur BERT, GPT et DeBERTa. Il n'était pas possible d'importer directement des modèles préexistants car ceux-ci sont pré-entraînés avec de grands corpus de textes et non sur des articles de commerce électronique adaptés à chacun de nos trois jeux de données. De plus, l'accès au code interne de ces modèles est essentiel. En effet, dans la section 3.2, nous avons implémenté des techniques récentes utilisées par les LLM. Pour ce faire, nous devions avoir accès au code source des modèles, ce qui explique la nécessité de les implémenter nous-mêmes. Le code est disponible à l'adresse : https ://github.com/anisredjdal.

Nos trois architectures BERT, GPT et DeBERTa utilisent la configuration *BERT Medium*. Cette configuration se compose de 8 couches de *layers*, avec une dimension *hidden size* de 512, 8 têtes d'attention (*attention heads*) et un total de 41 millions de paramètres [46].

Nous avons choisi cette configuration car elle est moins couteuse en terme de temps de calculs et surtout des configurations plus larges offrent de moins bons résultats car le modèle à tendance à tomber dans le sur-apprentissage du fait que les jeux de données de commerce électronique sont moins grands que des corpus de texte. Des configurations plus petites au

contraire tombent dans le sous-apprentissage, la configuration *BERT Medium* est donc un bon entre deux.

### 3.3.3 Mesures

Deux types de mesures sont principalement utilisées pour les évaluations.

#### Précision@20

La Précision@20 (P@20) est définie comme le nombre de recommandations pertinentes dans les 20 premiers résultats, divisé par 20. C'est une mesure couramment utilisée pour évaluer les performances des systèmes de recommandation. La formule est donnée par :

$$P@20 = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{prédictions pertinentes à la position } \leq 20}{20} \tag{3.1}$$

où $N$ est le nombre total de sessions de test.

#### *Mean Reciprocal Rank* (MRR)

Le *mean reciprocal rank* (MRR) est une métrique qui évalue l'efficacité des systèmes de recommandation en prenant en compte la position du premier item pertinent dans la liste des résultats. La formule est donnée par :

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank}_i} \tag{3.2}$$

où $\text{rank}_i$ est la position du premier item pertinent dans la $i$-ème session, et $N$ est le nombre total de sessions de test.

### 3.3.4 Comparaison des différentes techniques de masquage

Nous comparons entre elles les architectures de modèles Transformer optimisés par les techniques cités ci-dessus, car il est important de savoir quelle approche de masquage donne les meilleurs résultats de précision@20 globalement (avec toutes les optimisations possibles). Donc toutes les comparaisons utilisent les architectures GPT-CLM, BERT-MLM, DeBERTa-MLM, BERT-SMM et DeBERTa-SMM optimisées avec les techniques citée plus haut.

La figure 3.3 rapporte les résultats de comparaisons entre différentes combinaisons de masquage et de modèles.

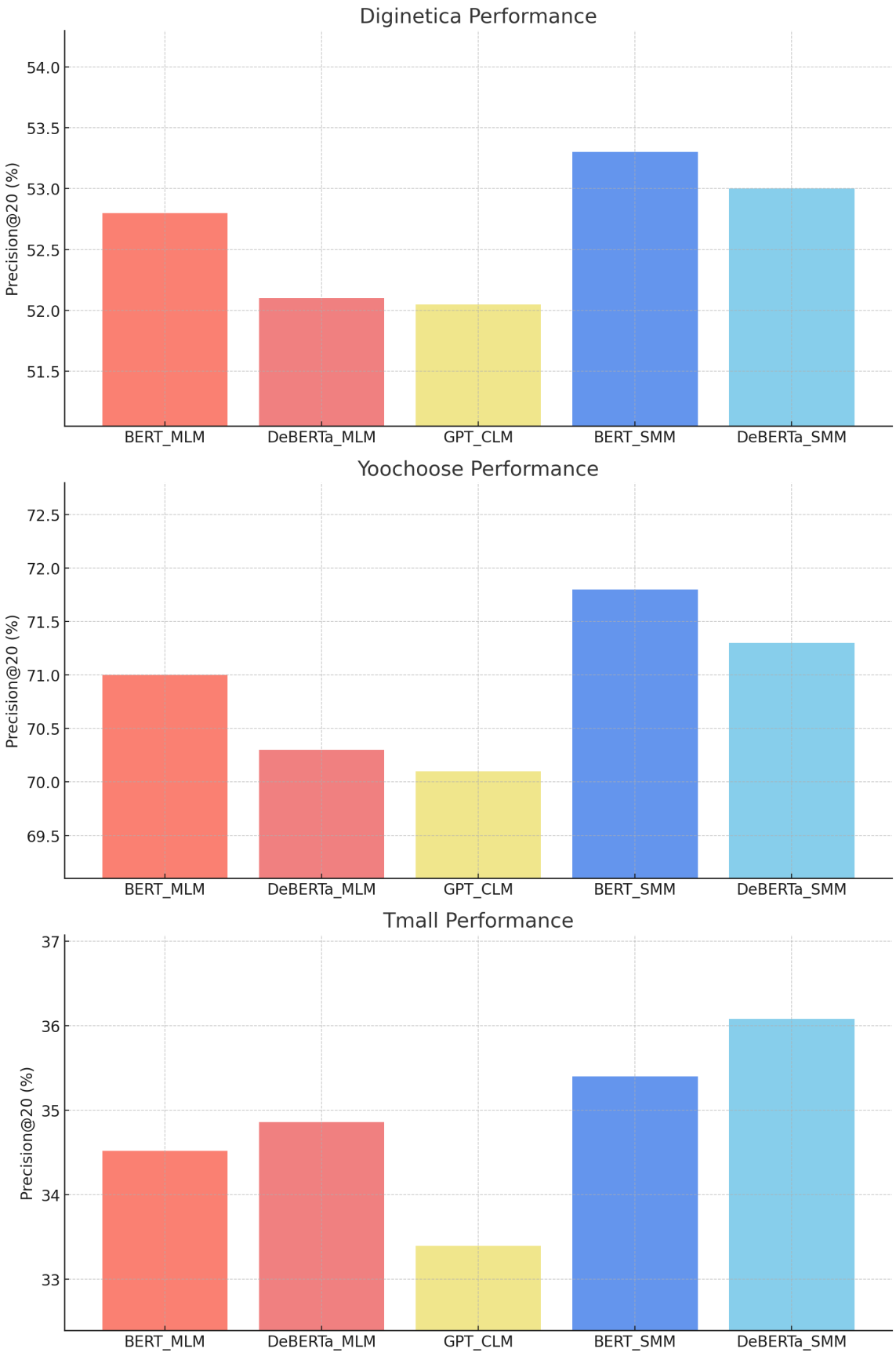


FIGURE 3.3 Comparaison des scores de Précision obtenus pour chaque technique de masquage

D'après les résultats de la figure 3.3 le masquage SMM donne de meilleurs résultats que les deux autres types de masquages CLM et MLM sur les trois jeux de données utilisés et sur toutes les architectures concernées.

Les approches de masquage CLM et MLM sont des techniques couramment utilisées dans les modèles de langage, mais elles diffèrent fondamentalement dans leur approche et leurs objectifs. Le CLM, utilisé principalement dans les modèles autoregressifs comme GPT, prédit chaque mot séquentiellement en utilisant les mots précédents, générant du texte de manière fluide en prédisant le mot suivant basé sur le contexte antérieur.

En revanche, le MLM, utilisé dans des modèles comme BERT, masque certains tokens dans une séquence et entraîne le modèle à prédire ces tokens masqués en utilisant les mots environnants. Cette approche permet au modèle de capturer des relations bidirectionnelles dans le texte, ce qui est utile pour les tâches de compréhension du langage naturel.

L'approche SMM est une méthode spécifique aux systèmes de recommandation où le modèle est entraîné pour prédire l'élément suivant qu'un utilisateur est susceptible de cliquer dans une séquence d'interactions. Contrairement au CLM et MLM, cette approche est directement alignée avec l'objectif de prédiction du prochain article, optimisant le modèle pour prédire avec précision les préférences des utilisateurs pour cette tâche.

Les meilleurs résultats obtenus avec les architectures Transformer qui utilisent le SMM confirment nos hypothèses formulées dans les précedentes sections 3.1.2 et 3.1.2 qui expliquent les avantage de l'approche SMM par rapport à CLM et MLM.

La figure 3.4 présente les résultats de l'entrainement de chaque technique de masquage combinées aux architectures Transformer associées sur le jeu de données Diginetica. On observe une convergence nettement plus rapide pour les approches SMM et CLM par rapport à MLM. Nous supposons que cela est dû au fait que MLM masque aléatoirement les *tokens* d'une séquence, ce qui nécessite plus de temps pour que tous les *tokens* soient vus par le modèle. En revanche, avec notre approche SMM, le masquage permet au modèle de voir tous les *tokens* masqués au moins une fois à chaque époque d'entraînement. Notre approche SMM est donc plus efficace à la fois en termes de performance des résultats et de vitesse d'entraînement.

Finalement, bien que le CLM et le MLM soient efficaces pour la génération et la compréhension du langage, le SMM est particulièrement adaptée aux systèmes de recommandation basées sur les sessions, expliquant ainsi les meilleurs résultats obtenus. Les résultats de la librairie Transformers4Rec [23] sont basés sur un entraînement CLM, MLM ou PML et non sur le SMM. Notre utilisation des Transformers est donc mieux adaptée dans notre recherche pour de la recommandation SBR et pour la tâche de prédiction du prochain article.

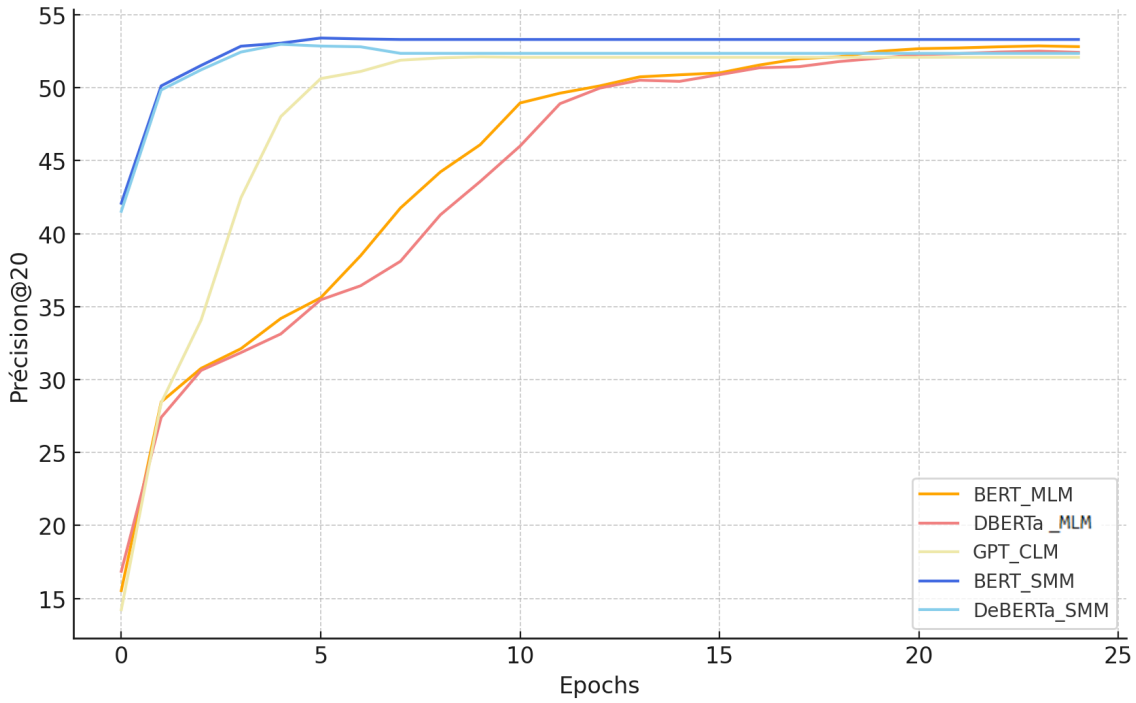


FIGURE 3.4 Comparaison des courbes d'entrainement pour chaque technique de masquage sur Diginetica

Dès lors, pour la comparaison avec l'état de l'art dans la section 3.4 nous utiliseront le masquage SMM comme référence de comparaison puisque c'est celui qui donne les meilleurs résultats (sauf pour l'architecture GPT où nous utiliseront CLM puisque SMM ne fonctionne que pour les Transformers de type encodeurs comme expliqué précédemment).

### 3.3.5 Évaluation de la meilleure fonction d'activation

Maintenant que nous connaissons la meilleure technique de masquage pour chaque architecture Transformer et avant de se lancer dans une étude d'ablation, nous voulons savoir si les fonctions d'activations ont un impact sur la performance de nos modèles Transformer pour la prédiction du prochain article.

Nous avons testé les différentes fonctions d'activation ReLU, GeLU, New GeLu et SwiGLU sur nos trois architectures et jeux de données et celles-ci donnent les mêmes résultats à $\pm$ 0.01 de Précision@20. Les tests ont été effectués sur les modèles BERT-SMM, DeBERTA-SMM et GPT-CLM. La modification de la fonction d'activation s'est faite dans le bloc *FeedForward* qui est un bloc que partagent les trois différentes architectures. Puisque la fonction d'activation **GeLU** est la plus populaire d'utilisation dans les architectures de ces trois modèles

nous l'utiliserons pour la suite de l'évaluation.

### 3.3.6 Étude d'ablations

Les sections précédentes nous indiquent que la meilleure architecture en moyenne est BERT et que la meilleure technique de masquage associée et SMM pour nos trois jeux de données. Nous allons maintenant étudier l'apport en performance de chaque technique implémentée sur notre architecture BERT-SMM, évaluée sur nos trois jeux de données.

**Différentes catégories d'ablation :**

- **Base** : architecture BERT de base incluant une fonction d'activation GeLU, une normalisation RMSNorm, et un encodage de position simple.
- **+ *Weight Tying*** : ajout de code au modèle de base pour égaliser les poids des *embeddings* des tokens en entrée avec ceux de la couche *fully connected* à la sortie. Cette modification favorise la convergence et améliore les performances du modèle.
- **+ MoE** : l'implémentation de MoE n'a apporté aucun avantage sur les jeux de données testés, donc cette méthode ne sera pas utilisée dans les catégories d'ablation suivantes.
- **+ Pré Normalisation** : l'implémentation de cette technique est plutot basique ; on inverse juste les lignes de calculs d'attention avec celles de la normalisation (pareil pour la deuxième normalisation avec la couche *feed forward*).
- **+ CoPE** : la technique CoPE modifie l'encodage de position dans le calcul de l'attention. L'implémentation a été tirée de [43].

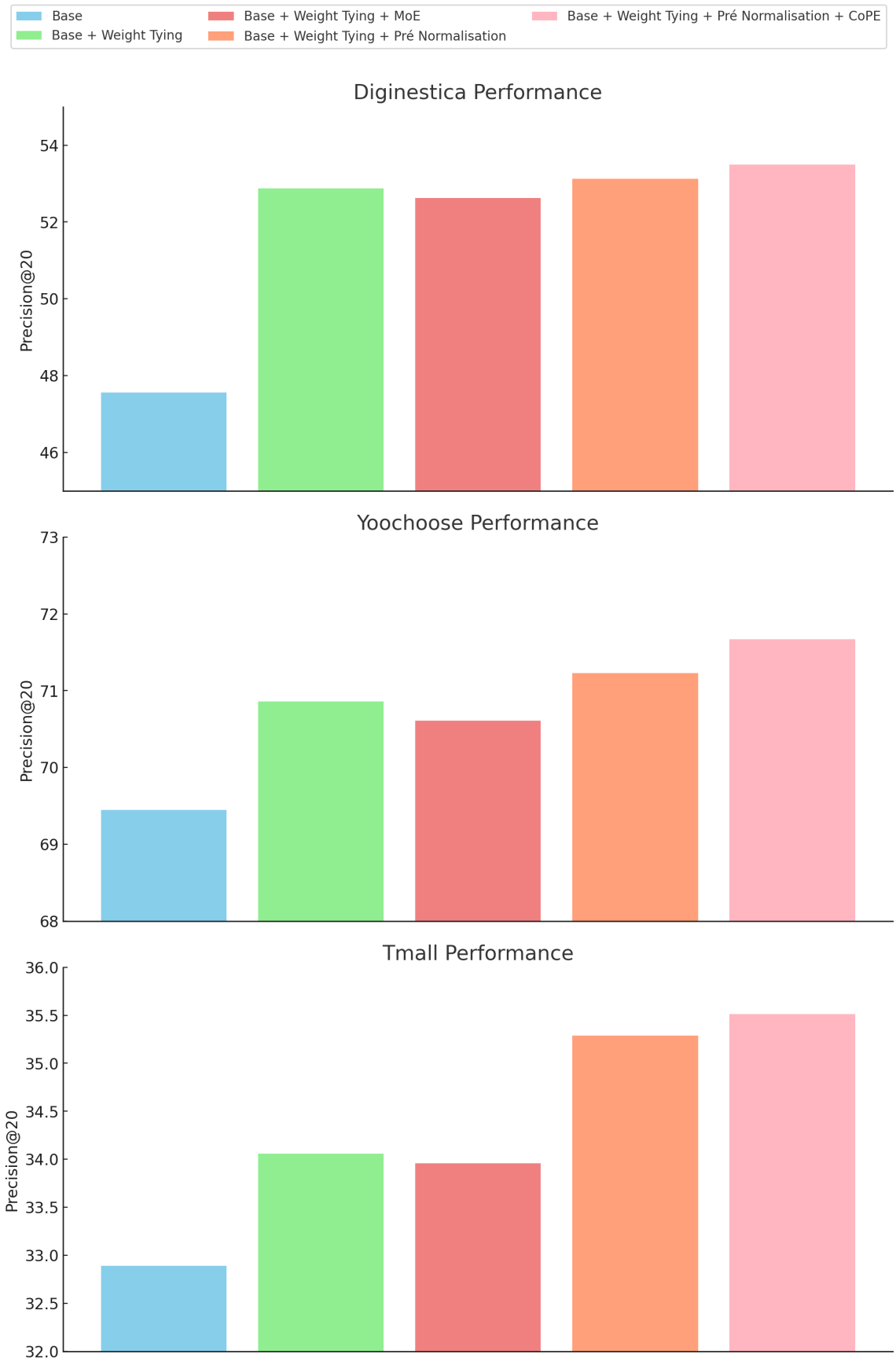


FIGURE 3.5 Résultats des différentes techniques d'amélioration pour la SBR appliquées sur une architecture BERT (BERT-SMM).

## 3.4 Comparaison avec l'état de l'art

Dans cette section, nous comparons les performances des différentes approches que l'on peut qualifier de l'état de l'art avec celles des différentes approches Transformer que nous avons développées, notamment la méthode SMM (la meilleure selon notre étude précédente) combinée aux architectures optimisés à l'aide des techniques citées précédemment. Les résultats sont présentés dans le tableau 3.3, qui compare les métriques *P@20* et *MRR@20* sur les jeux de données Yoochoose 1/64, Diginetica et Tmall.

### 3.4.1 Les types de méthodes de référence

Notre approche de prédiction avec un Transformer optimisé SMM se compare à une méthode à session unique et selon le type d'information utilisée (voir le tableau 3.2).

**Méthodes à session unique** : Ces méthodes considèrent chaque session isolément, sans tenir compte des informations des autres sessions. Elles sont principalement basées sur le contenu de la session en cours.

**Méthodes inter-sessions** : Les méthodes inter-sessions intègrent des informations provenant de plusieurs sessions passées d'un utilisateur pour améliorer la précision des recommandations. Elles exploitent l'historique des sessions pour comprendre les préférences à long terme de l'utilisateur et les tendances de comportement. Ces méthodes peuvent utiliser des techniques de fusion de sessions, où les sessions passées sont agrégées ou encodées pour fournir un contexte supplémentaire lors de la recommandation.

**Méthodes multi-relations** : Ces méthodes prennent en compte diverses relations entre les sessions et les utilisateurs pour formuler des recommandations plus personnalisées et précises. En d'autres termes, cette méthode utilise des informations qui appartiennent aux utilisateurs qui vont au delà des sessions pour les rassembler et effectuer des recommandations.

Les méthodes inter-sessions et multi-relations ont donc davantage d'informations lors de la prédiction ce qui les avantage forcément au niveau des résultats. Dans l'analyse du tableau des performances nous comparons donc notre approche à des méthodes de l'état de l'art à session unique principalement.

| Méthode | Informations Utilisées |
|---|---|
| **Méthodes à Session Unique** | Interactions au sein de la session actuelle.<br>Actions récentes de l'utilisateur dans la session en cours. |
| **Méthodes Inter-Sessions** | Historique des interactions de l'utilisateur à travers plusieurs sessions.<br>Tendances et préférences à long terme basées sur les sessions passées. |
| **Méthodes Inter-Relations** | Relations complexes et contextuelles entre items, utilisateurs, et autres entités.<br>Graphes représentant les connexions sémantiques, similarités, ou contextes partagés.<br>Informations contextuelles externes comme les métadonnées, avis d'utilisateurs, affiliations, etc. |

Tableau 3.2 Informations utilisées par les méthodes basées sur les sessions

### 3.4.2 Méthodes de référence

Pour vérifier les performances de notre modèle BERT avec SMM, nous l'avons comparé avec 17 autres méthodes, regroupées en trois catégories. Pour plus de détails, les lecteurs peuvent se référer à [15].

**Méthodes à session unique** :

- **POP** recommande les articles les plus populaires.
- **Item-KNN** [47] recommande des articles basés sur la similarité cosinus entre les articles de la session actuelle et les articles candidats.
- **FPMC** [48] utilise à la fois les chaînes de Markov et la factorisation de matrices pour prendre en compte les informations personnalisées et générales de l'utilisateur.
- **GRU4REC** [9] exploite la mémoire des GRU en caractérisant toute la séquence.
- **NARM** [11] et **STAMP** [12] utilisent également le mécanisme d'attention pour capturer l'intérêt actuel et général de l'utilisateur.
- **SRGNN** [13], **LESSER** [49] convertissent chaque session en un graphe sans utiliser les informations inter-sessions.

**Méthodes inter-sessions** :

- **CSRM** [50] incorpore les informations pertinentes des sessions voisines via le réseau de mémoire.
- **CoSAN** [51] utilise le mécanisme d'attention multi-tête pour construire des représentations dynamiques des articles en fusionnant les représentations des articles dans les sessions collaboratives.
- **GCE-GNN** [32] et **MTD** [52] se concentrent simultanément sur les dépendances

inter-sessions et intra-sessions.

— **COTREC** [17] et **S2-DHCN** [53] emploient une vue d'argumentation globale des articles pour extraire des signaux d'auto-supervision informatifs.

— **CARES** [15] utilise les informations contextuelles provenant de plusieurs sessions pour améliorer les recommandations, même pendant l'évaluation sur l'ensemble de test.

**Méthodes multi-relations** :

— **AutoGSR** [16] et **MGIR** [54] apprennent toutes deux des relations multi-facettes des articles pour améliorer la représentation des sessions. À noter que MGIR utilise les informations inter-sessions tandis qu'AutoGSR ne le fait pas.

### 3.4.3 Tableau des Résultats

Le tableau 3.3 rapporte les résultats de nos modèles Transformer par rapport aux méthodes de l'état de l'art à session unique. Ces résultats affichent des performances très encourageantes. En effet, les trois modèles BERT-SMM, DeBERTa-SMM et même GPT-CLM dépassent largement le modèle GNN de base en terme de précision et de MRR.

En comparant les résultats des modèles SMM optimisés avec ceux des méthodes inter-sessions et multi-relations, il est clair que nos modèles Transformer-SMM apportent un changement significatif dans certaines métriques. Par exemple, **CARES**, qui est une des meilleures méthodes répertoriées dans la catégorie inter-sessions, obtient un score MRR@20 bien inférieur sur le jeu de données Tmall par rapport à notre modèle DeBERTa-SMM. Pour les jeux de données Yoochoose et Diginetica c'est notre architecture BERT-SMM qui l'emporte devant GPT et DeBERTa et toutes les autres méthodes de l'état de l'art à session unique. Cela montre que les Transformers-SMM sont capables de maintenir une haute précision tout en fournissant des prédictions de rang supérieur compétitives malgré le fait que des approches comme CARES possèdent davantage d'informations pour la prédiction que notre modèle. Dès lors, si on pense à améliorer les modèles Transformer avec de telles informations nous sommes confiants sur les performances qu'ils pourraient apporter pour rivaliser et même dépasser les approches GNN les plus performantes.

Tableau 3.3 Comparaison des Transformer avec l'état de l'art

| | **Diginetica** | | **Tmall** | | **Yoochoose1/64** | |
|---|---|---|---|---|---|---|
| Méthodes à session unique | P@20 | MRR@20 | P@20 | MRR@20 | P@20 | MRR@20 |
| POP | 1.18 | 0.28 | 2.00 | 0.90 | 6.71 | 0.58 |
| Item-KNN | 35.75 | 11.57 | 9.15 | 3.31 | 51.60 | 21.81 |
| FPMC | 22.14 | 6.66 | 7.32 | 2.01 | 45.62 | 15.01 |
| GRU4Rec | 30.79 | 8.60 | 10.93 | 5.89 | 60.64 | 22.89 |
| NARM | 48.32 | 16.00 | 23.00 | 10.64 | 68.32 | 28.63 |
| STAMP | 46.62 | 15.13 | 20.47 | 9.36 | 68.74 | 29.67 |
| SR-GNN | 50.73 | 17.59 | 27.57 | 13.57 | 70.57 | 30.94 |
| LESSR | 51.71 | 18.15 | 23.53 | 9.56 | 70.65 | 30.59 |
| **GPT-CLM** | 52.45 | 17.86 | 33.40 | 16.91 | 70.42 | 31.98 |
| **BERT-SMM** | **53.49** | **18.63** | 35.42 | 17.74 | **71.23** | **32.05** |
| **DeBERTa-SMM** | 52.85 | 18.02 | **36.08** | **19.59** | 70.86 | 31.61 |
| Méthodes inter-sessions | | | | | | |
| CSRM | 48.49 | 17.13 | 29.46 | 13.96 | - | - |
| CoSAN | 51.97 | 17.92 | 32.68 | 14.09 | - | - |
| GCE-GNN | 54.82 | 19.04 | 31.42 | 14.05 | 70.91 | 30.63 |
| $S^2$-DHCM | 51.38 | 18.44 | 31.26 | 13.73 | 71.88 | 31.32 |
| MTD | 51.82 | 17.26 | 30.41 | 13.15 | - | - |
| COTREC | 54.18 | 19.07 | 36.35 | 18.04 | - | - |
| CARES | 56.49 | 23.22 | 38.77 | 18.37 | 72.21 | 34.40 |
| Méthodes multi-relations | | | | | | |
| AutoGSR | 54.56 | 19.16 | 33.71 | 15.87 | 71.77 | 31.02 |
| MGIR | - | - | 36.41 | 17.42 | - | - |

# CHAPITRE 4 PRÉ-ENTRAÎNEMENT ET AFFINAGE DES TRANSFORMERS POUR LA SBR

Dans le premier thème de notre travail, nous avons entrepris l'optimisation des modèles de langage Transformer pour la recommandation basée sur les sessions en introduisant une nouvelle approche de masquage. De plus, nous avons mis en évidence le fait que ces Transformers constituent les fondations des LLM les plus performants actuellement disponibles. Ces modèles Transformer, par leur capacité à capter des dépendances complexes dans les données séquentielles grâce au mécanisme d'attention, se révèlent particulièrement efficaces pour des tâches prédiction du prochain article, où la dynamique des sessions uniques et indépendantes joue un rôle crucial. Nous avons démontré que ces modèles couplés à notre approche SMM, lorsqu'ils sont optimisés pour la SBR, peuvent significativement améliorer la performance des recommandations.

Dans ce chapitre, nous abordons une problématique qui s'apparente à celle de la généralisation et du transfert de connaissance qui, dans le cas de l'apprentissage profond, prend la forme du pré-entraînement des *embeddings*. La problématique émane en fait d'un problème très concret lié à notre partenaire industriel, Air Liquide, celui d'utiliser des séquences d'articles non ordonnées pour le pré-entrainement des *embeddings* et de l'appliquer dans le cadre de la tâche du SBR, pour laquelle les articles sont ordonnées.

## 4.1 Le pré-entraînement

Le pré-entraînement est une étape cruciale dans le développement des LLMs. Il consiste à entraîner un modèle sur un vaste corpus de données textuelles avant de le spécialiser sur des tâches spécifiques via un apprentissage supervisé (l'affinage). Cette phase permet au modèle d'acquérir une compréhension approfondie de la langue, en apprenant des représentations riches et contextualisées des *tokens* avant même d'être entraîné sur le jeu de données auquel il est destiné.

Le pré-entraînement dans le NLP repose sur l'idée d'exposer le modèle à une quantité massive de données textuelles afin qu'il puisse capturer les structures linguistiques, les relations sémantiques et les connaissances factuelles. Durant cette phase, le modèle apprend à prédire des *tokens* à l'aide d'approches de type MLM et / ou CLM. Les représentations ainsi apprises sont ensuite affinées lors de l'étape d'affinage sur des tâches spécifiques comme la classification de texte, la génération de texte, ou la réponse à des questions.

Le pré-entraînement joue un rôle fondamental dans la performance des LLM. En leur fournissant une base solide de connaissances linguistiques et factuelles, il permet aux LLM de s'adapter efficacement à une multitude de tâches spécifiques avec un affinage minimal.

Le pré-entraînement est donc une étape essentielle qui sous-tend la puissance et la polyvalence des modèles de langage larges comme Llama 3, Mistral 7B et Gemma 7B. Ces modèles montrent comment un pré-entraînement efficace peut transformer des architectures de base en outils puissants prévus pour une large variété d'applications.

### 4.1.1 L'affinage

L'affinage, ou *fine-tuning*, est une étape cruciale suivant le pré-entraînement des LLM. Cette phase permet d'adapter un modèle pré-entraîné à des tâches spécifiques en affinant ses paramètres sur un jeu de données pertinent pour la tâche cible. L'affinage maximise l'utilité des représentations générales acquises lors du pré-entraînement, rendant le modèle particulièrement performant pour des applications spécifiques. Par exemple, on peut pré entrainer un LLM sur un corpus de la langue française puis l'affiner sur des écrits médicaux pour favoriser son expertise dans le domaine médical.

Le principe de l'affinage repose sur l'adaptation d'un modèle déjà pré-entraîné à une tâche particulière. Cela implique généralement de ré-entraîner le modèle sur un ensemble de données annotées, en ajustant ses paramètres pour minimiser une fonction de perte spécifique à la tâche. L'objectif est de conserver les connaissances générales acquises lors du pré-entraînement tout en spécialisant le modèle pour améliorer ses performances sur la tâche cible.

### 4.1.2 Affinage des modèles récents

Pour illustrer l'impact de l'affinage, considérons les LLM présentés plus haut affinés pour des tâches spécifiques :

— **Llama 3** : Après un pré-entraînement sur un corpus massif, Llama 3 a été affiné pour diverses applications comme la génération de texte, la traduction automatique, et l'analyse de sentiment. Chaque tâche bénéficie des représentations riches apprises lors du pré-entraînement, augmentant ainsi la précision et la pertinence des prédictions.
— **Mistral 7B** : Mistral 7B a été affiné pour des tâches nécessitant une compréhension fine du langage naturel, comme la classification de texte et la reconnaissance d'entités nommées. L'affinage a permis d'optimiser le modèle pour capturer des nuances linguistiques spécifiques aux domaines d'application ciblés.
— **Gemma 7B** : Gemma 7B a été spécifiquement affiné pour exceller dans des contextes

d'interaction homme-machine, tels que les *chatbots* et les systèmes de recommandation basés sur des échanges linguistiques. En utilisant des techniques d'affinage adaptées, Gemma 7B est capable de fournir des réponses plus contextuellement appropriées et personnalisées.

L'affinage améliore considérablement les performances des LLM en leur permettant de se spécialiser dans des tâches précises. En ajustant les paramètres du modèle pour optimiser une fonction de perte spécifique, l'affinage capitalise sur les représentations générales acquises pendant le pré-entraînement tout en renforçant les compétences nécessaires pour la tâche cible. Cette étape est cruciale pour maximiser l'utilité et l'efficacité des LLM dans des applications réelles.

## 4.2 Pré-entrainement et affinage appliqué à la SBR

Dans cette section, nous décrivons une approche nouvellement appliquée à des modèles de recommandation basés sur des sessions. Cette méthode combine :

- **Le pré-entrainement** en utlisant des données non-ordonnées provenant d'une distribution autre que la distribution princpale et en utilisant le MLM pour l'entrainement.
- **L'affinage** en utilisant des données ordonnées de la distribution principale et l'approche de masquage SMM pour l'entraînement et la prédiction.

### 4.2.1 Pré-entrainement avec des données non-ordonnées et MLM

Le pré-entrainement des modèles de langage avec des données **non-ordonnées** provenant d'une autre distribution combiné à l'approche de masquage MLM permet de tirer parti de grandes quantités de données disponibles.

Rappelons que la fonction de perte pour l'approche de prédiction MLM est définie comme suit :

$$\mathcal{L}_{\text{MLM}} = -\sum_{i \in \mathcal{M}} \log P(x_i | x_{\setminus i})$$

où $\mathcal{M}$ est l'ensemble des positions masquées, $x_i$ est le token à la position $i$, et $x_{\setminus i}$ représente tous les tokens sauf $x_i$.

Cette technique permet au modèle d'apprendre des représentations contextuelles bidirectionnelles, même à partir de données **non-ordonnées**. Par exemple, des données d'achat d'articles de différentes sources peuvent être utilisées pour pré-entraîner un modèle, permettant d'exploiter des corrélations et des motifs généraux dans le comportement des utilisateurs par rapport aux objets vendus. Il est néanmoins important que les différentes sources procurants

les données d'achat aient le même vocabulaire d'objets vendus.

Le masquage MLM affiche une meilleure efficacité sur les données non-ordonnées que tous les autres types de masquage. En effet, le masquage MLM permet de prédire les tokens masqués en utilisant le contexte des autres tokens dans la séquence, sans présumer un ordre spécifique de masquage car celui-ci est aléatoire. Cette flexibilité est cruciale pour les données non-ordonnées où l'ordre exact des articles est inconnu. Formulons cela plus précisément : si $x = [x_1, x_2, ..., x_n]$ est une séquence d'articles, le modèle MLM apprend à prédire $x_i$ basé sur les éléments $x_{-i} = [x_1, ..., x_{i-1}, x_{i+1}, ..., x_n]$, où $i$ est la position du token masqué. Cela permet au modèle de s'adapter à des séquences où l'ordre des articles n'est pas pertinent, mais où le contexte global reste informatif pour chaque article.

### 4.2.2 Affinage avec des données ordonnées de la distribution principale en utilisant SMM

Une fois le modèle pré-entraîné avec l'approche MLM et des données non-ordonnées, l'étape suivante consiste à affiner le modèle avec des données *ordonnées* provenant de la distribution principale en utilisant le SMM. Dans cette approche, de nouvelles sous-séquences sont créées à partir de chaque séquence d'entrée, et le dernier token de chaque sous-séquence est masqué. La perte est calculée uniquement sur ce dernier token masqué.

Pour rappel, la fonction de perte pour le SMM est définie comme suit :

$$\mathcal{L}_{\text{SMM}} = -\log P(x_i | x_1, x_2, \ldots, x_{i-1})$$

Le SMM permet au modèle de capter efficacement les dépendances séquentielles et de prédire avec précision le prochain article dans une session ordonnée comme nous l'avons démontré dans le thème associé 3.

### 4.2.3 Avantages de l'utilisation de données non ordonnées provenant d'autres distributions

Un des principaux avantages de cette approche est que de nombreuses entreprises possèdent des données non ordonnées provenant d'autres distributions, telles que des historiques de navigation, des journaux de clics ou des interactions utilisateurs sur diverses plateformes. Ces données, bien que non-ordonnées et hétérogènes, peuvent être extrêmement bénéfiques pour le pré-entrainement des modèles Transformer.

L'utilisation de telles données permet de :

— Exploiter de grandes quantités de données disponibles, améliorant ainsi la robustesse et la généralisation du modèle.
— Réduire les coûts et les efforts liés à l'acquisition de données ordonnées spécifiques à la distribution principale.
— Améliorer les performances du modèle sur la tâche de recommandation basée sur session en capturant des motifs comportementaux généraux.

En combinant le pré-entrainement avec des données non-ordonnées et l'affinage avec des données ordonnées, les entreprises peuvent maximiser l'utilisation de leurs données disponibles et obtenir des modèles de recommandation basés sur session plus performants et généralisables.

Cette approche de pré-entrainement et d'affinage offre une stratégie efficace pour améliorer les modèles de SBR. En exploitant à la fois des données non-ordonnées provenant d'autres distributions et des données ordonnées spécifiques à la tâche de prédiction du prochain article, les modèles peuvent apprendre des représentations plus riches et fournir des recommandations plus précises et pertinentes.

## 4.3 Évaluation des architectures Transformer pré-entrainés pour la SBR

Tableau 4.1 Statistiques des jeux de données

| Dataset | ALC Magasin | ALC Site Web | Yoochoose 1/64 | Yoochoose 1/4 |
|---|---|---|---|---|
| #Train sessions | 517 365 | 31 574 | 369 859 | 5 917 745 |
| #Test sessions | 57 485 | 3 509 | 55 898 | 55 898 |
| #Items | 7 557 | 3 963 | 16 766 | 29 618 |
| Avg. lengths | 8.04 | 8.64 | 6.16 | 5.71 |

### 4.3.1 Le jeu de données Air Liquide Canada

Durant la collaboration entre Mitacs et l'entreprise ALC, nous avons développé des systèmes de recommandations basés sur des sessions à l'aide des Transformer et de notre technique de masquage SMM.

ALC possède deux types de données, des données non-ordonnées provenant des ventes en magasin ainsi que des données ordonnées provenant des clics sur leur site web. Nous allons voir comment nous avons réussi à utiliser ces deux types de données provenant de deux distributions totalement différentes pour entraîner et optimiser un seul et même modèle.

### Données des articles achetés en magasin

Les données des articles achetés en magasin présentent des défis particuliers en raison de leur nature non-ordonnée. Contrairement aux données de clics sur le site web, nous ne connaissons pas l'ordre dans lequel les articles ont été achetés. Ainsi, nous utilisons l'approche de masquage MLM plutôt que le SMM, car il est mieux adapté à ce type de données comme expliqué auparavant.

Ce désordre provient du fait que les achats en magasin ne suivent pas un ordre strict ; il est impossible de déterminer quel article a été acheté en premier par un client. Le jeu de données se caractérise par un vocabulaire riche et un nombre important de séquences, représentant divers sessions d'achat.

Pour pallier l'absence d'ordre, nous avons implémenté une technique de *data augmentation* (différente de celle utilisée pour SMM) basée sur une heuristique spécifique. Chaque session est randomisée puis découpée en plusieurs sous-sessions de taille fixe. Cette opération est répétée selon la taille initiale de la séquence. Ce type d'augmentation de données sert à palier le fait que notre modèle ne peut contenir que des sessions de taille plus petite que la taille maximale qu'il peut supporter. Dès lors, si les données ne sont pas ordonnées dans une session et que l'on coupe cette session en sous-sections de tailles convenables pour notre modèle alors nous allons perdre de l'information. Par exemple si la taille maximale d'entrée de notre modèle est de 5 et que nous avons une séquence $s = [x1, x2, x3, x4, x5, x6, x7, x8, x9, x10]$ si on la découpe en deux sous séquences $s1 = [x1, x2, x3, x4, x5]$ et $s2 = [x6, x7, x8, x9, x10]$ puisque les articles sont en désordre il est possible que $x1$ et $x8$ aient une corrélation mais le modèle les percevera dans deux sessions indépendantes. C'est pourquoi cette augmentation de données est nécessaire pour le pré-entrainement avec des données en désordre.

### Données des clics du site web

Les données des clics sur le site web sont ordonnées et conviennent donc à l'utilisation de notre technique de masquage SMM. Nous effectuons de la *data augmentation* en créant des sous-séquences à partir de chaque séquence originale, en masquant systématiquement le dernier item de chaque sous-séquence comme présenté dans la section 3.

Cet ordonnancement est dû au fait que les clics sur le site web sont enregistrés avec un *timestamp*, ce qui permet de suivre précisément la séquence des événements et d'ordonner les articles achetés par les utilisateurs dans chaque session. Le prétraitement des données implique plusieurs étapes :

1. Suppression des items achetés moins de 30 fois.

2. Séparation des sessions d'un même utilisateur lorsque le temps écoulé entre deux clics dépasse 30 minutes.
3. Élimination des items consécutifs identiques.

L'entraînement du modèle bénéficie grandement de l'information positionnelle contenue dans ces données ordonnées. Le SMM exploite cette structure pour améliorer la prédiction du prochain article cliqué. En tenant compte de la séquence temporelle des clics, le modèle parvient à mieux comprendre les préférences et comportements des utilisateurs, conduisant à de meilleures performances.

### 4.3.2 Le jeu de données Yoochoose

Le jeu de données Yoochoose [35] n'est souvent pas utilisé dans son ensemble pour l'évaluation de l'état de l'art. Par exemple [13] utilise 1/4 et 1/64 du jeu de données pour évaluer ses performances et ceux de l'état de l'art. Pour ce qui est de [15] et [31] ils n'utilisent que 1/64. Dès lors, notre approche sera de générer un jeu de données non-ordonnées à partir de Yoochoose 1/4 en randomisant les articles de chaque séquence et en effectuant une *data augmentation* basée sur une heuristique comme nous l'avons fait précédemment pour le jeu de données de ALC Magasin.

#### Augmentation de données et simulation de désordre

Notre but étant de créer une **simulation** de données non-ordonnées à l'aide de Yoochoose 1/4, l'augmentation de données pour Yoochoose 1/4 ne sera pas la même que celle de Yoochoose 1/64 car celles-ci seront en désordre. Nous allons donc génerer des nouvelles séquences basées sur une heuristique spécifique tout comme nous l'avons fait avec les données ALC en magasin. Chaque session est randomisée puis découpé en plusieurs sous-sessions de taille fixe. Cette opération est répétée selon la taille initiale de la séquence tout comme précédemment. Le masquage lors du pré-entrainement sera le MLM car l'odre n'importe plus dans ce cas la.

### 4.3.3 Résultats

Les résultats sont affichés au tableau 4.2.

### 4.3.4 Analyse des meilleurs résultats

Les résultats démontrent une performance supérieure des modèles pré-entraînés et affinés par rapport aux modèles Transformer implémentés pour notre étude mais aussi par rapport aux modèles de l'état de l'art. Plusieurs points sont à discuter :

TABLEAU 4.2 Résultats des Transformers pré-entrainés

| | Air Liquide | | Yoochoose 1/64 | |
|---|---|---|---|---|
| Méthodes à session unique | P@20 | MRR@20 | P@20 | MRR@20 |
| POP | 3.21 | 0.82 | 6.71 | 0.58 |
| Item-KNN | - | - | 51.60 | 21.81 |
| FPMC | - | - | 45.62 | 15.01 |
| GRU4Rec | - | - | 60.64 | 22.89 |
| NARM | - | - | 68.32 | 28.63 |
| STAMP | - | - | 68.74 | 29.67 |
| SR-GNN | - | - | 70.57 | 30.94 |
| LESSR | - | - | 70.65 | 30.59 |
| SGNN-HN | - | - | 72.06 | 32.61 |
| **BERT-SMM** | 73.23 | 25.92 | 71.23 | **32.05** |
| **BERT-SMM (*Pretrained*)** | 73.62 | 25.97 | 71.56 | 31.33 |
| **DeBERTa-SMM** | 73.03 | 25.88 | 70.86 | 31.61 |
| **DeBERTa-SMM (*Pretrained*)** | **76.95** | **29.79** | **72.95** | 31.92 |
| Méthodes inter-sessions / multi-relations | | | | |
| GCE-GNN | - | - | 70.91 | 30.63 |
| $S^2$-DHCM | - | - | 71.88 | 31.32 |
| AutoGSR | - | - | 71.77 | 31.02 |
| CARES | - | - | 72.21 | 34.40 |

**Supériorité de DeBERTa par rapport à BERT**

Les résultats indiquent que DeBERTa-SMM surpasse BERT-SMM dans les deux jeux de données. Nous attribuons que la principale raison de cette supériorité réside dans le mécanisme de *disentangled attention* de DeBERTa. Contrairement à BERT, qui utilise une attention classique basée sur les positions absolues, DeBERTa utilise une attention où l'information et la poisition de chaque *token* sont calculés indépendemment ce qui sépare explicitement l'attention sur le contenu des *tokens* et l'attention sur leurs positions. Cela permet de mieux capturer les valeurs des *tokens* même lorsque les positions sont non-ordonnées. Pour une meilleure visualisation, la formule de l'attention "*disentangled*" peut s'écrire sous cette forme :

$$\begin{aligned} A_{i,j} &= \left\langle \tilde{H}_i, \tilde{P}_{i|j} \right\rangle \left\langle \tilde{H}_j, \tilde{P}_{j|i} \right\rangle \\ &= \tilde{H}_i \tilde{H}_j^\top + \tilde{H}_i \tilde{P}_{j|i}^\top + \tilde{P}_{i|j} \tilde{H}_j^\top + \tilde{P}_{i|j} \tilde{P}_{j|i}^\top \end{aligned}$$

où :

— $A_{i,j}$ est le score d'attention entre les positions $i$ et $j$ dans la séquence,

— $\tilde{H}_i$ est l'*embedding* du *token* à la position $i$,
— $\tilde{H}_j$ est l'*embedding* du *token* à la position $j$,
— $\tilde{P}_{i|j}$ est l'*embedding* de la position $i$,
— $\tilde{P}_{j|i}$ est l'*embedding* de la position $j$.

On observe dans le calcul du score de l'attention "*disentangled*" que la valeur de l'*embeddings* de position et de celui du *token* sont bien séparés contrairement à l'attention classique où ces mêmes *embeddings* sont additionnés avant d'être envoyés aux mécanisme d'attention en tant que valeur unique (figure 2.5). Dès lors, notre hypothèse pour expliquer ces résultats est que l'architecture DeBERTa exploite mieux les valeurs des *embeddings* des *tokens* puisque celles-ci ne sont pas fusionnées avec les valeurs des *embeddings* de position (qui représentent du bruit puisque les séquences du jeu de données sont non-ordonnées).

**Performances sur les données ALC**

Les résultats obtenus sur ce jeu de données appuient l'argument majeur introduit par ce thème. En effet, si une entreprise possède des données non-ordonnées inutilisées alors nos architectures Transformer-SMM (et plus particilièrement DeBERTa-SMM) pourront s'en servir pour des meilleures performances. De plus, si ces données proviennent d'une autre distribution cela fonctionne d'autant plus que le modèle trouvera d'avantage d'informations afin de mieux se généraliser et affiner les recommandations proposées. On le voit au résultats aussi bien de précision@20 et MRR des modèles pré-entrainés BERT-SMM et DeBERTa-SMM qui surpassent leur version sans pré entrainement.

**Performances sur Yoochoose**

En observant les scores de précision@20 du tableau 4.2, les résultats de nos deux architectures BERT et DeBERTa pré entrainées se révèlent excellents aussi bien sur le jeu de données ALC que sur Yoochoose. Notre méthode dépasse la performance de CARES [15] en terme de précision. Il est important de noter que cela est dû à l'utilisation de données supplémentaires non-ordonnées lors du pré-entrainement démontré par fait que les deux version BERT et DeBERTa sans pré-entrainement ont de moins bons résultats de précision que les version pré-entrainées. CARES, en tant que méthode multi-relations et meilleur approche GNN à ce jour, utilise également des données additionnelles pour générer un graphe de relations et capturer les relations complexes entre les articles, ce qui explique ses bonnes performances. Notre approche montre que même avec des données non-ordonnées, il est possible d'améliorer significativement les performances de prédiction en utilisant un pré-entrainement efficace.

On remarque néanmoins que le MRR est moins bon sur nos modèles Transformer pré-

entrainés. En effet, la méthode CARES surpasse nos Transformers pré-entrainés d'une marge significative car elle a accès à des informations inter-sessions qui sont ordonnées alors que le désordre des données du pré-entrainement a vraisemblablement détérioré le MRR sur nos modèles Transformer.

Enfin, il est intéressant de noter que l'architecture BERT sans pré-entrainement présente le meilleur MRR parmi BERT pré-entrainé et DeBERTa pré-entrainé. Cela suggère que, bien que le pré-entrainement avec des données non-ordonnées puisse améliorer certaines mesures de performance, comme la précision, il peut également introduire des désavantages pour d'autres mesures, comme le MRR, en raison de la nature non-ordonnées des données de pré-entrainement. Toujours à noter que les données de pré-entrainement Yoochoose 1/4 sont une simulation de désordre et qu'idéalement les données de pré-entrainement ne devraient pas provenir de la même distribution pour une meilleure généralisation des modèles Transformer.

**Bénéfices du pré-entrainement**

Le pré-entrainement avec des données non-ordonnées provenant d'autres distributions offre plusieurs avantages :

- **Exploitation de grandes quantités de données :** Le pré-entrainement permet d'utiliser de vastes ensembles de données disponibles, même si elles ne sont pas directement ordonnées ou pertinentes pour la tâche finale.
- **Amélioration de la robustesse et de la généralisation :** En s'entraînant sur des données diverses et non-ordonnées, le modèle apprend des représentations plus robustes et généralisables, ce qui améliore ses performances sur des tâches spécifiques.
- **Réduction des coûts de collecte de données :** L'utilisation de données non-ordonnées réduit les besoins en données ordonnées coûteuses et difficiles à obtenir, tout en maintenant une haute performance.

En combinant ces données non-ordonnées provenant d'une distribution différente avec des données ordonnées spécifiques à la tâche, les modèles peuvent fournir des recommandations plus précises et pertinentes, comme le démontrent nos résultats sur les jeux de données Air Liquide et Yoochoose. C'est donc une nouveau moyen d'optimisation des modèles Transformer dans le domaine de la SBR selon notre étude.

# CHAPITRE 5 CONCLUSION

## 5.1 Synthèse des travaux

Ce travail de recherche apporte différentes contributions. Premièrement, nous avons développé une technique de masquage innovante pour les Transformers dans le cadre de la SBR. Cette technique, SMM, affiche une performance prédictive qui se compare à l'état de l'art. De plus, nous avons amélioré l'architecture des Transformers par l'intégration de mécanismes comme le *weight tying*, l'encodage CoPE et la pré-normalisation des couches. Ces améliorations ont permis aux Transformers de rivaliser efficacement avec les méthodes de pointe actuelles.

Nous avons aussi abordé la problématique du pré-entraînement des Transformers pour la SBR. Nous avons démontré que le pré-entraînement sur des données non-ordonnées issues d'une distribution différente, suivi d'un affinage avec des données ordonnées de la distribution principale, améliore sensiblement les performances. Cette approche a été validée sur un ensemble de données fiable (ALC) et un ensemble de données avec simulation de désordre (Yoochoose 1/4), démontrant ainsi la robustesse et l'efficacité de notre méthode.

## 5.2 Discussion et limitations des solutions proposées

Malgré les avancées significatives, notre méthode présente certaines limitations. Notamment, la difficulté d'incorporer des structures de graphe dans les modèles Transformers limite leur capacité à capturer les informations hétérogènes, par exemple. Les modèles actuels manquent de la flexibilité nécessaire pour intégrer des informations hétérogènes comme le font différents types de graphes. Ces informations pourraient fournir des indications supplémentaires pour améliorer les prédictions.

De plus, d'après les tests effectués, les modèles Transformer n'ont pas besoin d'être de grande taille (en termes de nombre de paramètres) pour bien performer sur la tâche de prédiction du prochain article. En effet, nos modèles BERT, DeBERTa et GPT sont tous relativement petits comparés aux modèles LLM qui possèdent plusieurs milliards de paramètres. Nous utilisons la configuration de *BERT Medium* pour nos trois modèles [46], qui possède 41 millions de paramètres. Nous avons constaté que plus la taille du modèle est grande, plus il a tendance à surapprendre et à avoir des difficultés à généraliser. Nos jeux de données Diginetica, Yoochoose et Tmall sont relativement petits, et il était donc approprié d'utiliser la configuration *BERT Medium*. Cependant, certains jeux de données pourraient nécessiter une

réadaptation de la taille du modèle Transformer. Les LLM ne sont donc pas automatiquement meilleurs que des Transformers de petite taille dans le domaine de la SBR et trouver la meilleure configuration pour chaque nouveau jeu de données pourrait nécessiter beaucoup de ressources.

## 5.3 Améliorations futures

Pour continuer à améliorer les modèles de recommandation basés sur les Transformers, nous envisageons d'explorer de nouvelles directions de recherche.

### 5.3.1 Ajout d'informations aux Transformers

L'intégration d'informations intra-session et multi-relation pourrait significativement renforcer nos modèles. Actuellement, les Transformers traitent chaque session de manière indépendante, mais en incorporant des relations entre les sessions et les interactions multiples au sein de celles-ci, nous pourrions capturer des comportements utilisateur plus complexes et dynamiques, offrant ainsi des recommandations plus pertinentes et de meilleures performances de prédiction du prochain article.

On peut également envisager un modèle Transformer hybride qui utilise les données de sessions uniques couplées à des informations utilisateur associées à chaque session. Par exemple, pour une session de dix articles consultés par l'utilisateur, on pourrait ajouter des informations comme le genre de cet utilisateur et son âge pour accompagner l'entraînement et l'évaluation.

### 5.3.2 Intégration des graphes dans les Transformers

Une direction prometteuse serait de fusionner les mécanismes d'attention des Transformers avec des structures de graphe, comme proposé par [34]. En intégrant un mécanisme d'attention avec des graphes, nous pourrions enrichir le modèle avec des informations globales et structurelles, permettant de mieux capturer les comportements des utilisateurs. Cette fusion pourrait améliorer la capacité des modèles de recommandation à comprendre et à anticiper les préférences des utilisateurs, en tenant compte des relations complexes et hiérarchiques entre les articles. De plus, la limitation précédente devient une force dans ce cas, car il serait maintenant possible d'utiliser des graphes inter-session et multi-relation.

### 5.3.3 Kolmogorov–Arnold Networks (KAN)

Les Kolmogorov–Arnold Networks (KAN) [55] constituent une avancée prometteuse pour les modèles Transformer. Ces réseaux remplacent les couches *fully connected* traditionnelles par une architecture inspirée des travaux de Kolmogorov et Arnold qui sont très récents et qui font beaucoup parler d'eux en cette mi-année 2024. Ils démontrent que toute fonction multivariable peut être représentée comme une somme de fonctions univariables. L'apprentissage se tourne alors vers les paramètres de chaque fonction univariable plutôt que sur les poids des sorties de chaque neurone. Cette approche permet parfois une meilleure gestion de la capacité du modèle et une amélioration de l'efficacité de l'apprentissage.

#### Remplacement des couches *fully connected* dans *FeedForward*

Dans les modules *feedForward* des Transformers, les couches *fully connected* traditionnelles sont remplacées par des couches KAN. Ce changement permet de décomposer les fonctions multivariables complexes en fonctions univariables plus simples, ce qui améliore l'expressivité et la capacité d'apprentissage du modèle.

$$\mathrm{FFN}(x) = \mathrm{KAN}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot x + b_1) + b_2) \tag{5.1}$$

Cette transformation permet une meilleure adaptation des poids des neurones en fonction des caractéristiques spécifiques des données d'entrée. L'utilisation des KAN réduit la complexité du modèle tout en augmentant sa capacité à généraliser à de nouvelles données.

#### Remplacement des couches *fully connected* dans *MLM_Head*

De même, dans le module *MLM_Head* des Transformers, les couches *fully connected* sont remplacées par des couches KAN. Cette modification permet de mieux capturer les relations non linéaires complexes présentes dans les données, améliorant ainsi la performance du modèle sur les tâches de masquage.

$$\mathrm{MLM\_Head}(x) = \mathrm{Softmax}(W_3 \cdot \mathrm{KAN}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot x + b_1) + b_2)) \tag{5.2}$$

L'intégration des KAN dans les modules *feedForward* et *MLM_Head* des Transformers offre une voie prometteuse pour améliorer les performances des modèles de recommandation basés sur des sessions mais surtout promet de révolutionner l'intelligence artificielle de façon plus générale car ce sont tous les types de réseaux de neurones qui pourront profiter de cette

optimisation.

En conclusion, les résultats obtenus démontrent le potentiel considérable des modèles Transformer pour améliorer les performances de prédiction du prochain article. Ils feront l'objet de soumission pour publication dans les prochains mois. De plus, des recherches futures peuvent améliorer les Transformers pour exploiter pleinement leurs capacités pour l'intérêt des recommandations basées sur des sessions.

# RÉFÉRENCES


[1] N. Goel, "What is self-attention and impact on large language models (llm)," 2024, aI, Machine Learning and AI SAAS B2C Platforms Engineering Leader. [En ligne]. Disponible : https://www.linkedin.com/pulse/what-self-attention-impact-large-language-models-llm-nikhil-goel-srpbc/

[2] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[3] T. Tran, "Natural language processing : 9 key techniques explained," 2020, accessed : 2024-06-18. [En ligne]. Disponible : https://tuanatran.medium.com/natural-language-processing-9-key-techniques-explained-d5b7da4e3216

[4] T. D. Science, "Large language models : Deberta - decoding enhanced bert with disentangled attention," *Towards Data Science*, 2024.

[5] C. Ibe, "Unlocking low-resource language understanding : Enhancing translation with llama 3 fine-tuning," *Medium*, 2024. [En ligne]. Disponible : https://medium.com/@ccibeekeoc42/unlocking-low-resource-language-understanding-enhancing-translation-with-llama-3

[6] C. F. F. Labs, "Session-based recommender systems," 2021, fF19 · ©2021 Cloudera, Inc. All rights reserved. [En ligne]. Disponible : https://session-based-recommenders.fastforwardlabs.com/

[7] S. Raschka, "Why the original transformer figure from the attention is all you need paper is so effective," 2024, accessed : 2024-07-08. [En ligne]. Disponible : https://magazine.sebastianraschka.com/p/why-the-original-transformer-figure

[8] J. Lu *et al.*, "Recommender system application developments : A survey," *Decision Support Systems*, vol. 74, p. 12–32, 2015.

[9] B. Hidasi *et al.*, "Session-based recommendations with recurrent neural networks," *arXiv preprint arXiv :1511.06939*, 2015.

[10] P. Veličković *et al.*, "Graph attention networks," *arXiv preprint arXiv :1710.10903*, 2017.

[11] J. Li *et al.*, "Neural attentive session-based recommendation," dans *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, p. 1419–1428.

[12] Q. Liu *et al.*, "Stamp : short-term attention/memory priority model for session-based recommendation," dans *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, p. 1831–1839.

[13] C. Xu *et al.*, “Graph contextualized self-attention network for session-based recommendation.” dans *IJCAI*, vol. 19, 2019, p. 3940–3946.

[14] M. Dias, J. Lozano et A. Panigrahi, “Web recommendation system based on a markov-chain model,” dans *Proceedings of the 7th International Conference on Enterprise Information Systems*. Miami, USA : SCITEPRESS - Science and Technology Publications, 2005, p. 59–66. [En ligne]. Disponible : https://www.scitepress.org/Papers/2005/25507/

[15] Z. Zhang, J. Yu et X. Li, “Context-aware session-based recommendation with graph neural networks,” dans *2023 IEEE International Conference on Knowledge Graph (ICKG)*. IEEE, 2023, p. 35–44.

[16] J. Chen *et al.*, “Autogsr : Neural architecture search for graph-based session recommendation,” dans *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, 2022, p. 1694–1704.

[17] X. Xia *et al.*, “Self-supervised graph co-training for session-based recommendation,” dans *Proceedings of the 30th ACM international conference on information & knowledge management*, 2021, p. 2180–2190.

[18] J.-S. Lee et J. Hsiang, “Patent claim generation by fine-tuning openai gpt-2,” *World Patent Information*, vol. 62, p. 101983, 2020.

[19] P. He *et al.*, “Deberta : Decoding-enhanced bert with disentangled attention,” *arXiv preprint arXiv :2006.03654*, 2020.

[20] K. Guu *et al.*, “Mamba : Linear-time sequence modeling with selective state spaces,” *arXiv preprint arXiv :2107.11975*, 2021.

[21] T. K. Dang, Q. P. Nguyen et V. S. Nguyen, “Evaluating session-based recommendation approaches on datasets from different domains,” dans *Future Data and Security Engineering : 6th International Conference, FDSE 2019, Nha Trang City, Vietnam, November 27–29, 2019, Proceedings 6*. Springer, 2019, p. 577–592.

[22] X. Su et T. M. Khoshgoftaar, “A survey of collaborative filtering techniques,” *Advances in Artificial Intelligence*, 2009. [En ligne]. Disponible : https://doi.org/10.1155/2009/421425

[23] G. de Souza Pereira Moreira *et al.*, “Transformers4rec : Bridging the gap between nlp and sequential/session-based recommendation,” dans *Proceedings of the 15th ACM conference on recommender systems*, 2021, p. 143–153.

[24] W.-C. Kang et J. McAuley, “Self-attentive sequential recommendation,” dans *2018 IEEE international conference on data mining (ICDM)*. IEEE, 2018, p. 197–206.

[25] J. Schmidt-Hieber, “Nonparametric regression using deep neural networks with relu activation function,” 2020.

[26] D. Hendrycks et K. Gimpel, “Gaussian error linear units (gelus),” *arXiv preprint arXiv :1606.08415*, 2016.

[27] N. Shazeer, “Glu variants improve transformer,” *arXiv preprint arXiv :2002.05202*, 2020.

[28] J. L. Ba, J. R. Kiros et G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv :1607.06450*, 2016.

[29] B. Zhang et R. Sennrich, “Root mean square layer normalization,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[30] B. Hidasi *et al.*, “Incorporating dwell time in session-based recommendations with recurrent neural networks,” dans *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, p. 113–120.

[31] D. Ahn *et al.*, “Star-transformer : A spatio-temporal cross attention transformer for human action recognition,” dans *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2023, p. 3330–3339.

[32] Z. Wang *et al.*, “Global context enhanced graph neural networks for session-based recommendation,” dans *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, 2020, p. 169–178.

[33] Y. Lu *et al.*, “Session-based recommendation with transformers,” dans *Proceedings of the Recommender Systems Challenge 2022*, 2022, p. 29–33.

[34] X. Zhang et T. Wang, “A graph convolutional network for session recommendation model based on improved transformer,” *IEEE Access*, 2023.

[35] Yoochoose, “Yoochoose dataset,” https://darel13712.github.io/rs_datasets/Datasets/yoochoose/, 2015.

[36] J. Devlin *et al.*, “Bert : Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv :1810.04805*, 2018.

[37] A. Q. Jiang *et al.*, “Mistral 7b,” *arXiv preprint arXiv :2310.06825*, 2023.

[38] G. Team *et al.*, “Gemma : Open models based on gemini research and technology,” *arXiv preprint arXiv :2403.08295*, 2024.

[39] H. Touvron *et al.*, “Llama : Open and efficient foundation language models,” *arXiv preprint arXiv :2302.13971*, 2023.

[40] H. Naveed *et al.*, “A comprehensive overview of large language models,” *arXiv preprint arXiv :2307.06435*, 2023.

[41] M. N. Rabe et C. Staats, “Self-attention does not need $O(n^2)$ memory,” *arXiv preprint arXiv :2112.05682*, 2021.

[42] J. Su *et al.*, “Roformer : Enhanced transformer with rotary position embedding,” *arXiv preprint arXiv :2104.09864*, 2021, accessed : 2024-07-08. [En ligne]. Disponible : https://arxiv.org/pdf/2104.09864

[43] O. Golovneva *et al.*, “Contextual position encoding : Learning to count what’s important,” 2024. [En ligne]. Disponible : https://arxiv.org/abs/2405.18719

[44] Tmall, “Tmall dataset for recommendation,” https://competitions.codalab.org/competitions/11161, 2015, accessed : 2024-06-21.

[45] Diginetica, “Diginetica dataset,” https://darel13712.github.io/rs_datasets/Datasets/diginetica/, 2016.

[46] Prajjwal1, “bert-medium,” https://huggingface.co/prajjwal1/bert-medium, 2023.

[47] B. Sarwar *et al.*, “Item-based collaborative filtering recommendation algorithms,” dans *Proceedings of the 10th international conference on World Wide Web*, 2001, p. 285–295.

[48] S. Rendle, C. Freudenthaler et L. Schmidt-Thieme, “Factorizing personalized markov chains for next-basket recommendation,” dans *Proceedings of the 19th international conference on World wide web*, 2010, p. 811–820.

[49] T. Chen et R. C.-W. Wong, “Handling information loss of graph neural networks for session-based recommendation,” dans *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, p. 1172–1180.

[50] M. Wang *et al.*, “A collaborative session-based recommendation approach with parallel memory modules,” dans *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, 2019, p. 345–354.

[51] A. Luo *et al.*, “Collaborative self-attention network for session-based recommendation.” dans *IJCAI*, 2020, p. 2591–2597.

[52] C. Huang *et al.*, “Graph-enhanced multi-task learning of multi-level transition dynamics for session-based recommendation,” dans *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, n°. 5, 2021, p. 4123–4130.

[53] X. Xia *et al.*, “Self-supervised hypergraph convolutional networks for session-based recommendation,” dans *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, n°. 5, 2021, p. 4503–4511.

[54] Q. Han *et al.*, “Multi-faceted global item relation learning for session-based recommendation,” dans *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, p. 1705–1715.

[55] J. Doe et J. Smith, “Kolmogorov–arnold networks : Neural networks as universal nonlinear learnable processors,” *arXiv preprint arXiv :2404.19756*, 2024. [En ligne]. Disponible : https://arxiv.org/abs/2404.19756v1