PGCOMP - Programa de Pós-Graduação em Ciência da Computação
Universidade Federal da Bahia (UFBA)
Av. Milton Santos, s/n - Ondina
Salvador, BA, Brasil, 40170-110

https://pgcomp.ufba.br
pgcomp@ufba.br

Recommender systems (RecSys) enhance information retrieval efficiency across different domains by delivering personalized content that aligns with user preferences. These systems address user-item relationships through methods like matrix factorization and graph attention networks (GAT). Despite advancements in accuracy, existing approaches often narrowly focus on predictive performance while neglecting the broader utility of confidence estimation. This estimation is crucial for quantifying the certainty behind recommendations, particularly in situations where a balance between risk and reward is required. RecSys can mitigate uncertainties stemming from data noise and model limitations by utilizing confidence. Existing approaches to confidence integration face critical limitations. Non-parametric techniques, such as neural network-based probabilistic calibration, remain confined to classification tasks, failing to address regression scenarios, including rating prediction and listwise learn-to-rank. Many confidence models operate independently of core recommendation processes, limiting their adaptability and calibration impact. Notably, the literature neglects the integration of confidence into GAT-based models. Additionally, the literature lacks experimental evaluation of different distribution-based methods. Therefore, this study proposes an experimental evaluation of previous distribution-based methods and explores a suitable confidence integration in GAT-based models. We evaluate four prior solutions in terms of rating prediction accuracy, ranking accuracy, and confidence correlation with error. These solutions and our proposal are evaluated in public datasets from varying contexts and characteristics. Results reveal that distribution-based confidence integration often harms models' accuracy and leaves room for improvement regarding the correlation between confidence and error. Although these findings also hold for our method, it still achieves superior performance compared to all prior solutions and shows promising results in terms of negative confidence–error correlation. Furthermore, as a second part of this study, we propose and evaluate the integration of confidence into embedding models for learn-to-rank methods. This proposal and its baselines are also evaluated across various public datasets, using different ranking metrics, and the correlation with confidence and error. The results reveal that both proposed methods consistently demonstrate competitive rank performances and even outperform the baselines in some datasets. Specifically, the proposed confidence integration for rating prediction achieved improvements of at least 58.16% in ranking metrics on the Amazon Movies and TVs, 34.94% on the Jester Joke, and 42.98% on the MovieLens dataset. Additionally, we observed a cubic polynomial relationship between confidence and error in this latter solution.

Keywords: Recommender Systems, Confidence, Uncertainty

MSC | 209 | 2026

Integrating Confidence into Embedding-Based Models for Learn-to-Rank in Recommender Systems

Joel Machado Pires

UFBA

Integrating Confidence into Embedding-Based Models for Learn-to-Rank in Recommender Systems

Joel Machado Pires

Dissertação de Mestrado

Universidade Federal da Bahia

Programa de Pós-Graduação em Ciência da Computação

January | 2026

Universidade Federal da Bahia
Instituto de Computação

Programa de Pós-Graduação em Ciência da Computação

INTEGRATING CONFIDENCE INTO EMBEDDING-BASED MODELS FOR LEARN-TO-RANK IN RECOMMENDER SYSTEMS

Joel Machado Pires

DISSERTAÇÃO DE MESTRADO

Salvador
20 January 2026

JOEL MACHADO PIRES

INTEGRATING CONFIDENCE INTO EMBEDDING-BASED MODELS FOR LEARN-TO-RANK IN RECOMMENDER SYSTEMS

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Frederico Araújo Durão

Salvador
20 January 2026

Ficha catalográfica elaborada pela Biblioteca Universitária de Ciências
e Tecnologias Prof. Omar Catunda, SIBI – UFBA.

P667 Pires, Joel Machado

Integrating confidence into embedding-based models for learn-to-rank in recommender systems / Joe Machado Pires. – Salvador, 2026.

88 f.

Orientador: Prof. Dr. Frederico Araújo Durão

Dissertação (Mestrado) – Universidade Federal da Bahia. Instituto de Computação, 2026.

1. Recommender Systems. 2. Confidence. 3. Reliability. I. Durão, Frederico Araújo. II. Universidade Federal da Bahia. III. Título.

CDU: 004.65:004.738.5

TERMO DE APROVAÇÃO

JOEL MACHADO PIRES

INTEGRATING CONFIDENCE INTO EMBEDDING-BASED MODELS FOR LEARN-TO-RANK IN RECOMMENDER SYSTEMS

Esta Dissertação de Mestrado foi julgada adequada à obtenção do título de Mestre em Ciência da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia.

Salvador, 20 January 2026

Prof. Dr. Frederico Araújo Durão
Federal University of Bahia

Documento assinado digitalmente
gov.br
ANDRE LEVI ZANON
Data: 09/02/2026 10:15:20-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. André Levi Zanon
Insight Research Ireland Centre for Data Analytics

Documento assinado digitalmente
gov.br
YURI DE ALMEIDA MALHEIROS BARBOSA
Data: 09/02/2026 09:55:54-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Yuri de Almeida Malheiros Barbosa
Federal University of Paraíba

ACKNOWLEDGEMENTS

The Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES – Brazil) supported this research through a scholarship granted under Code 88887.941435/2024-00.

I express my sincere gratitude to my advisor, Prof. Fred Durão, for his guidance, support, and patience throughout the development of this work. I also thank the faculty and colleagues of the Institute of Computing, whose insights and discussions were invaluable contributions to this research. In particular, I would like to thank Eduardo Ferreira, Mayki Oliveira, Diego Correa, and other colleagues who supported me along the way.

I am deeply grateful to my family for their constant encouragement and support during this journey. Thank you to my father, José Pires; my mother, Maria Pires; my sister, Aline Pires; and my grandmother, Maria Terezinha Pires. I also extend my appreciation to Rege Pires, Mário Pires, Eliana Nunes, Yan Nunes, Paula Silva, and Tainane Nunes for their support. Finally, I would like to thank Talita Silva, who accompanied me through different stages of this process; your presence at those times helped make this achievement possible.

To my friends, thank you as well. In particular, to Roberta Paixão and Faith Jerry. Each of you contributed energy and motivation that helped me reach this moment.

RESUMO

Os sistemas de recomendação aumentam a eficiência da recuperação de informações em diversos domínios e oferecem conteúdo personalizado alinhado às preferências do usuário. Eles extraem relações usuário-item por meio de métodos como fatoração de matrizes e redes de atenção em grafos (GAT). A literatura apresenta avanços em precisão para tarefas de regressão e ordenação, mas ainda se concentra principalmente no desempenho preditivo, com pouca atenção à utilidade mais ampla da estimativa de confiança. A estimativa de confiança é essencial para quantificar a certeza das recomendações, especialmente quando é necessário equilibrar risco e recompensa. Ela permite mitigar incertezas decorrentes de ruído nos dados e limitações do modelo. No entanto, abordagens existentes enfrentam limitações importantes. Técnicas não paramétricas, como calibração probabilística com redes neurais, permanecem restritas a tarefas de classificação e não contemplam cenários de regressão, como previsão de notas ou aprendizado para ranqueamento. Além disso, muitos modelos de confiança operam de forma independente do modelo principal de recomendação, o que prejudica a capacidade de refletir adequadamente o nível real de certeza. A literatura também negligencia a integração de confiança em modelos baseados em GAT e carece de avaliações experimentais comparativas de métodos baseados em distribuição. Diante dessas lacunas, este estudo propõe uma avaliação experimental de métodos anteriores baseados em distribuição e investiga uma integração adequada de confiança em modelos baseados em GAT. Avaliamos quatro soluções existentes em termos de precisão na previsão de notas, precisão de ordenação e correlação entre confiança e erro, utilizando conjuntos de dados públicos com diferentes características. Os resultados mostram que a integração de confiança baseada em distribuição frequentemente reduz a precisão e ainda oferece espaço para melhorias na correlação confiança-erro. Embora essas conclusões também se apliquem ao nosso método, ele apresenta desempenho superior ao de todas as soluções anteriores e obtém resultados promissores de correlação negativa entre confiança e erro. Na segunda parte do estudo, propomos e avaliamos a integração de confiança em métodos de aprendizado de ranqueamento. Essa proposta, bem como os modelos de referência, é avaliada em múltiplos conjuntos de dados e métricas de ordenação. Especificamente, a integração de confiança proposta para modelos de previsão de classificação alcançou melhorias de pelo menos 58,1636% nas métricas de classificação no conjunto de dados Amazon Movies and TVs, 34,94% no conjunto de dados Jester Joke e 42,9882% no conjunto de dados MovieLens. O método proposto para aprendizado de classificação não prejudica significativamente o desempenho do modelo, apresentando resultados competitivos ou melhores em todos os conjuntos de dados. Além disso, observamos uma relação polinomial cúbica entre confiança e erro nesta última solução.

Palavras-chave: Sistemas de recomendação, Confiança, Incerteza

ABSTRACT

Recommender systems (RecSys) enhance information retrieval efficiency across different domains by delivering personalized content that aligns with user preferences. These systems address user-item relationships through methods like matrix factorization and graph attention networks (GAT). Despite advancements in accuracy, existing approaches often narrowly focus on predictive performance while neglecting the broader utility of confidence estimation. This estimation is crucial for quantifying the certainty behind recommendations, particularly in situations where a balance between risk and reward is required. RecSys can mitigate uncertainties stemming from data noise and model limitations by utilizing confidence. Existing approaches to confidence integration face critical limitations. Non-parametric techniques, such as neural network-based probabilistic calibration, remain confined to classification tasks, failing to address regression scenarios, including rating prediction and listwise learn-to-rank. Many confidence models operate independently of core recommendation processes, limiting their adaptability and calibration impact. Notably, the literature neglects the integration of confidence into GAT-based models. Additionally, the literature lacks experimental evaluation of different distribution-based methods. Therefore, this study proposes an experimental evaluation of previous distribution-based methods and explores a suitable confidence integration in GAT-based models. We evaluate four prior solutions in terms of rating prediction accuracy, ranking accuracy, and confidence correlation with error. These solutions and our proposal are evaluated in public datasets from varying contexts and characteristics. Results reveal that distribution-based confidence integration often harms models' accuracy and leaves room for improvement regarding the correlation between confidence and error. Although these findings also hold for our method, it still achieves superior performance compared to all prior solutions and shows promising results in terms of negative confidence–error correlation. Furthermore, as a second part of this study, we propose and evaluate the integration of confidence into embedding models for learn-to-rank methods. This proposal and its baselines are also evaluated across various public datasets, using different ranking metrics, and the correlation with confidence and error. The results reveal that both proposed methods consistently demonstrate competitive rank performances and even outperform the baselines in some datasets. Specifically, the proposed confidence integration for rating prediction achieved improvements of at least 58.16% in ranking metrics on the Amazon Movies and TVs, 34.94% on the Jester Joke, and 42.98% on the MovieLens dataset. Additionally, we observed a cubic polynomial relationship between confidence and error in this latter solution.

Keywords: Recommender Systems, Confidence Estimation, Uncertainty

CONTENTS

ABBREVIATIONS

RecSys	recommendation systems	1
NN	neural networks	5
GNN	graph neural networks	2
CF	collaborative filtering	1
CBF	content-based filtering	1
RNN	recurrent neural networks	2
CNN	convolutional neural networks	2
NDCG	normalized discounted cumulative gain	7
MAP	mean average precision	7
MSE	mean squared error	22
MAE	mean absolute error	18
RMSE	root mean squared Error	7
MF	matrix factorization	1
GAT	graph attention networks	2
ML	machine learning	21
DF	demographic filtering	12
ID	identification	47
BPR	bayesian personalized ranking	6
CPMF	confidence-aware probabilistic matrix factorization	5
CBPMF	confidence-aware bayesian probabilistic matrix factorization	5
LBD	learned beta distributions	5
CDF	cumulative distribution function	51
PRGAT	probabilistic recommender graph attention networks	8
NLL	negative log likelihood loss	60
ID	identification	47
OrdRec	OrdRec	54
MRR	mean reciprocal rank	65
BPR	bayesian personalized ranking	6

Chapter 1

This chapter provides context for the subject addressed in this dissertation, defines the motivations, and introduces the central problems of the field scope. Next, we present the objectives that guide the analyses conducted throughout the study. We also outline the methodology adopted to achieve these objectives. Finally, we outline the dissertation's structure to guide the reader through the subsequent chapters.

INTRODUCTION

Recommendation systems (RecSys) enhances information retrieval efficiency across various domains. It has applications in diverse fields such as search engines, marketing, streaming platforms, social media, smart cities, and automation (ANDRADE-RUIZ et al., 2024; ZHAROVA et al., 2024). For example, search engines present results that closely align with the user's query. Similarly, RecSys suggests content that aligns with the user's preferences in the platforms. In this context, RecSys functions as an intermediary, benefiting both users and businesses. For users, RecSys enhances satisfaction, optimizes time utilization, offers personalized recommendations, facilitates exploration of new content, and potentially leads to cost savings. For businesses, RecSys boosts sales, improves user retention, and fosters loyalty (XU et al., 2024; RICCI et al., 2011).

RecSys can be broadly classified into collaborative filtering (CF), content-based filtering (CBF), or hybrid systems. It handles the relationship between users and items. Where items can be any content that can be recommended to the users. CF methods aim to discern underlying user or item interactions to identify groups to which a target user or item belongs. Conversely, CBF methods group items based on their attributes. Hybrid systems amalgamate elements of both approaches by leveraging historical user behavior or content information (LI et al., 2024; SHAIKH et al., 2024).

In the realm of personalized RecSys, CF poses a significant challenge in modeling the relationships between users and other users or items. Approaches such as matrix factorization (MF) and deep learning (DL)-based models utilize user interactions with items to capture this intricate relationship between users and items. However, these approaches often encounter several challenges, including the cold start, high data sparsity, high computational resource demands, and modeling challenges, which augment their predictive error.

The cold start problem arises in scenarios of extreme data sparsity, wherein users have no prior or few interactions recorded in the system. Data sparsity refers to the proportion

of missing or zero values in a dataset. One strategy to mitigate data sparsity issues in the models involves incorporating side information related to users and item metadata into the representations of users or items (GAO et al., 2023). This allows the model to leverage that information, rather than solely the interactions, to learn the relationship between users and items. For example, content-based filtering groups items similar to a target item based on their attributes and recommends them to users who liked the target item. In particular, clustering algorithms such as hierarchical clustering, k-means, and Gaussian mixture models are widely employed to group users or items based on their attributes (NETO; COSTA; MANZATO, 2018a; BAHRANI et al., 2024; DENG et al., 2018). In the context of mitigating data sparsity, these algorithms leverage personal information from users and metadata from items. Metadata includes title/name, description, number of interactions, and any other attribute that an item can have. While these strategies help alleviate data sparsity issues, the challenge remains in effectively modeling the complex relationships between users and items.

Matrix factorization is a widely adopted CF approach that models the relationship between users and items, where the product of user and item representations in latent space is the simplest one. Despite its effectiveness, the literature continues to explore other modeling solutions and combinations of MF with other models. For instance, applying convolutional neural networks (CNN)s captures spatial relationships and provides representations at different levels. Sequential methods, such as recurrent neural networks (RNN)s and transformer-based models, capture sequential relationships (YOU et al., 2019; LI et al., 2020). Autoencoder models, which can use simple feedforward, CNNs, and sequential, can yield encoded representations of user-item vectors and reconstruct the user-item interactions matrix with all possible ratings, making them suitable for both rating prediction and clustering tasks (ZHANG et al., 2019). Graph attention networks (GAT)s have been introduced as a novel approach that captures nonlinear and higher-level relationships between users and items. In particular, GATs have demonstrated effectiveness and robustness in modeling user-item relationships through graph representations and encoding them as continuous latent vectors, which allows for integration with other data encodings (WU et al., 2022; PENG; SUGIYAMA; MINE, 2024). GAT is introduced as a variation of graph neural networks (GNN) that uses the attention concept (VELIčKOVIć et al., 2018). Although these solutions help improve the recommendation system's accuracy by handling the relationship between users and items, they still fail to answer whether the system's recommendations are reliable.

This issue can arise from the stochastic nature of user preferences and the sparsity of interaction data that introduce uncertainty into these predictions (PAPADOPOULOS; EDWARDS; MURRAY, 2001; KOREN; SILL, 2011; WANG et al., 2018a). This uncertainty may lead to unreliable recommendations, especially in high-risk domains or for decision-critical applications. The presence of unreliable recommendations can be exacerbated when the model fails to provide any indication of prediction reliability. To address this, confidence estimation has been proposed as a way to quantify the reliability of model predictions, providing an inverse measure of prediction uncertainty (MESAS; BELLOGÍN, 2020; KOREN; SILL, 2011; KNYAZEV; OOSTERHUIS, 2023).

1.1 MOTIVATION

The confidence level in RecSys reflects the system's certainty regarding its recommendations and can be estimated at various stages of the RecSys process. To illustrate, consider a stock price recommendation system where the model predicts a 95% probability that a stock will yield a 30% profit but with only 50% confidence. Another option may have an 89% probability of achieving a 15% profit with 90% confidence. Here, the probability represents the model's prediction, while confidence quantifies how certain the model is about that probability. Conversely, we can assess the intrinsic uncertainty of a model's prediction, which directly relates to its confidence level.

Model uncertainty is related to the errors present in the dataset, often due to errors in data collection or human error, as well as model error. The uncertainty in the dataset is caused by its noise, often present in the interaction registration process. For example, in RecSys, human error is often encountered when a user evaluates a product that does not accurately reflect their taste due to a flawed metric interpretation, resulting in data noise. Meanwhile, the model noise is caused by the model's inability to learn the dataset patterns (PAPADOPOULOS; EDWARDS; MURRAY, 2001). In some probabilistic models, confidence can be interpreted as the complement of uncertainty when uncertainty is explicitly defined as a probability measure. However, in many machine learning models, confidence often refers to the model's internal belief in its prediction, such as softmax output in classification and probability scores in recommendation systems. Uncertainty can refer to various factors, such as model variance, entropy, or epistemic and aleatoric uncertainty. While confidence and uncertainty are inversely related in many cases, they are not always strict complements. Still, their estimation plays a similar role in RecSys. Confidence or uncertainty estimation can be broadly categorized into parametric and non-parametric approaches. Parametric methods, such as modeling user's item relevance or confidence estimation using Normal, Beta, or Gamma distributions, provide structured frameworks for incorporating uncertainty (KNYAZEV; OOSTERHUIS, 2023; KWEON, 2024a; WANG et al., 2018a). We also refer to these methods as distribution-based.

Recommendations, along with their confidence estimations, help users make more informed decisions and build trust in the system. For instance, stock recommendation systems serve as a mechanism for harmonizing the trade-off between risk and return. In an educational recommendation system, confidence estimation can significantly enhance decision-making for both students and teachers. Confidence scores in the course or resource recommendations can help students gauge how well a suggested material aligns with their learning needs. If a platform recommends a specific textbook or online course with high confidence, the student can trust that it is a strong match based on their learning history and performance. From a teacher's perspective, confidence levels in recommendations for grouping students or selecting teaching approaches can improve instructional strategies. This transparency allows both students and educators to make more informed choices. By embedding confidence quantification into the recommendation framework, such systems enhance decision-making rigor and cultivate user trust. Confidence is also applicable in other contexts of RecSys, such as in data analysis and model improvement. In this regard, confidence can play a role in user segmentation and

grouping, refining how CF models identify relationships between users.

For example, a utilization of confidence is for the system to determine a user's group membership with a certain confidence level in CF. This can serve as a weighted similarity to define how strongly a user belongs to a group. Then, it can use this information to estimate ratings for items based on user-group interactions, where users with low confidence should contribute less to the group characteristics. From another perspective, RecSys can utilize confidence levels to select appropriate neighbors for a target user or item, filtering out those whose confidence is insufficient (NETO; COSTA; MANZATO, 2018b; GOHARI; ALIEE; HAGHIGHI, 2018). This confidence estimation can be utilized to refine the boundaries and overlapping groups. Beyond improving user segmentation, confidence estimation plays a crucial role in multiple aspects of RecSys, extending beyond evaluating individual predictions to enhancing overall system performance. It provides a more robust framework for assessing RecSys models, particularly when multiple models exhibit similar performance levels. By prioritizing models with higher overall confidence in their predictions, organizations can ensure more reliable recommendations. Additionally, confidence serves as a valuable tool for co-training, where high-confidence predictions can be leveraged to refine and retrain models, as demonstrated by Shaikh et al. (2024) through a data augmentation approach (SHAIKH et al., 2024). Moreover, confidence estimation can enhance novelty and diversity in recommendations. By identifying items with high scores but low confidence, or vice versa, the system can balance exploration and exploitation, thereby introducing users to a broader range of relevant content (HASSAN, 2019). Confidence information also improves ranking strategies by filtering low-confidence recommendations, adjusting their presentation, and reordering rankings to optimize both reliability and user experience. Ultimately, integrating confidence-aware mechanisms into RecSys enhances decision-making, fosters user trust, and strengthens the overall effectiveness of recommendation models. Conversely, failing to incorporate confidence or uncertainty modeling can introduce significant challenges, undermining both the reliability of recommendations and the user experience.

This omission prevents users from distinguishing between high-confidence and low-confidence recommendations, leading to poor decision-making and a decline in trust in the system (MESAS; BELLOGÍN, 2020; KWEON, 2024b). Without confidence estimation, users may be exposed to inaccurate recommendations without any indication of their reliability, resulting in frustration and disengagement. From a model performance perspective, the absence of confidence-aware mechanisms limits the system's ability to assess and refine its predictions. Clustering-based RecSys, for example, may incorporate noisy users into groups, reducing the quality of extracted features and leading to suboptimal recommendations. Additionally, training data may contain significant noise, which can negatively impact model learning and generalization. Since all predictions are treated as equally certain, the system fails to account for cases where the model exhibits high uncertainty, which can compromise the quality of recommendations. Moreover, ranking strategies remain suboptimal without confidence-aware adjustments. The system may prioritize recommendations without considering uncertainty, leading to less effective rankings that fail to maximize user satisfaction. This results in a ranking that, while functional, does not fully leverage the model's potential to deliver superior ordering.

Ultimately, ignoring confidence and uncertainty reduces the effectiveness of RecSys and undermines user trust and system reliability. As a result, user engagement and satisfaction may decline, leading to decreased sales on platforms utilizing this RecSys. At the very least, user satisfaction would fall short of its potential compared to a system incorporating confidence or uncertainty estimation. Consequently, the company may miss out on potential revenue opportunities. The literature has been realizing these issues, where we can evidence some contributions for confidence or uncertainty estimation (MOON et al., 2020).

1.2 PROBLEM

Among non-parametric methods for confidence integration in CF models, neural networks (NN)-based approaches are prevalent. Some studies have introduced probabilistic calibration techniques, using the model's output as a confidence estimation (MESAS; BELLOGÍN, 2020). Other works—such as those by Gohari et al. (2018) and Himabindu et al. (2018) employ heuristic methods to estimate confidence and use it for output calibration. However, these models are inherently limited to classification tasks, as their outputs are constrained to a probability range (e.g., logistic regression in [0,1]), making them unsuitable for regression or learn-to-rank problems where predictions represent continuous values, such as ratings. Furthermore, some confidence or uncertainty estimation techniques are modeled separately from the main prediction process, meaning they do not influence the model's output. This lack of integration can lead to inconsistencies in confidence estimation, reduced adaptability, and missed opportunities for improving the model's reliability and calibration.

The parametric or distribution-based methods often impose a distribution on the model's outcomes, such as a Gaussian, Gamma, or Beta distribution. Previous work has explored these confidence modeling strategies primarily in MF-based models, while GAT-based models remain underexplored in this context (KOREN; SILL, 2011; KNYAZEV; OOSTERHUIS, 2023; WANG et al., 2018a). Additionally, despite growing attention to confidence integration in RecSys, existing methods have not been holistically benchmarked, limiting our understanding of their strengths, weaknesses, and applicability. In particular, the literature lacks a clear understanding of how previous confidence modeling techniques can be adapted to GAT-based architectures. For instance, OrdRec by Koren and Sill (2011), and confidence-aware probabilistic matrix factorization (CPMF) and confidence-aware bayesian probabilistic matrix factorization (CBPMF) by Wang et al. (2018) model uncertainty using predefined distributions, such as the Gamma and Gaussian distributions. Learned beta distributions (LBD), proposed by Knyazev and Oosterhuis (2023), follows a similar approach, imposing a Beta distribution. This reveals a trend towards distribution-based methods.

However, none of these approaches provides confidence estimation tailored for GAT-based models, which remain underrepresented in this area. GAT-based models are considered state of the art in recommender systems due to their ability to effectively process graph-structured data. This modeling strategy aligns well with recommendation tasks, as recommendations are generated based on relationships among entities, which are nat-

urally represented as graphs (WANG et al., 2018b; WANG et al., 2019; TAO et al., 2020; FENG et al., 2019).

Moreover, methods such as those proposed in (KOREN; SILL, 2011; WANG et al., 2018a) often degrade accuracy, while the approach in (KNYAZEV; OOSTERHUIS, 2023) suffers from convergence issues. Yet, the confidence quality, such as negative correlation with the model's error, of these methods still leaves room for improvements (COSCRATO; BRIDGE, 2023). Finally, the literature lacks experimental evaluation of prior distribution-based methods, for instance, by evaluating their relationship with prediction error across different model types. In particular, does not demonstrate whether these estimates are reliable.

Furthermore, most early solutions focus on confidence methods for rating prediction, especially those aforementioned. Attempts to extend confidence estimation to listwise or pairwise learn-to-rank are scarce. Besides, post-processing solutions (ZHANG; GUO; CHEN, 2016) and heuristic-based approaches (MESAS; BELLOGÍN, 2020) introduce either high computational costs or model-specific assumptions. Deep learning–based uncertainty modeling (WANG; KADIOĞLU, 2023) advances the field by distinguishing between epistemic and aleatoric uncertainty; however, its inference overhead and restriction to neural architectures limit its broader applicability. To the best of our knowledge, no prior work has proposed learning confidence estimates for the relevance scores produced by pairwise learn-to-rank models.

Therefore, we aim to experimentally evaluate prior distribution–based methods to identify existing technical gaps. Based on this analysis, we propose a method that mitigates the identified limitations by introducing an additional confidence output into embedding-based rating prediction models, such as MF and GAT, fully integrated into both the model architecture and the learning process. In addition, we introduce a second proposal with similar characteristics that addresses the lack of confidence integration in listwise learning-to-rank methods.

1.3 OBJECTIVE AND CONTRIBUTIONS

In this context, the objectives of this work are:

- Address the lack of experimental evaluation and integration strategies for confidence-aware recommender systems by analyzing and proposing methods that incorporate confidence into both MF and GAT models.

  (OB1) Experimental evaluation of prior distribution-based methods regarding error and rank metrics, as well as confidence-error relationship and confidence distributions.

  (OB2) Propose and evaluate a method that integrates confidence in CF embedding-based models without harming accuracy and mitigating confidence-error low correlation for the rating prediction task.

- Derive a calibrated, uncertainty-aware learning-to-rank framework by reformulating the bayesian personalized ranking (BPR) method, to enable the joint optimization

of relevance scores and predictive variance across embedding-based architectures without compromising computational efficiency.

(OB3) Evaluate the confidence-error correlation of the proposal.

(OB4) Assess whether adding confidence improves robustness and consistency across metrics and cutoffs.

By pursuing the objectives associated with the first proposal, we make the following contributions:

- Benchmarking of four distribution-based confidence-aware models - OrdRec, CPMF, CBPMF, and LBD — on three public datasets (Amazon Movies and TVs, Jester-Joke, and MovieLens) using standard metrics for rating prediction (root mean squared Error (RMSE)) and ranking (normalized discounted cumulative gain (NDCG), mean average precision (MAP)).
- Proposing a model architecture with confidence integration.
- Analyzing the relationship between model confidence and prediction error across datasets
- Providing insights into confidence calibration and the limitations of existing confidence-aware approaches.

By addressing the objectives of the second proposal, we make the following contributions:

- A unified uncertainty-aware listwise formulation using a Gaussian CDF.
- Confidence-aware variants that preserve or improve ranking performance across four datasets and two model families.
- Empirical evidence that uncertainty reflects predictive reliability with strong negative Pearson correlations between confidence and BPR error.
- Demonstrate uncertainty magnitudes and correlations reflect the model's reliability and dataset properties such as sparsity, interaction independence, and domain complexity.
- A cross-domain evaluation demonstrating general applicability for listwise learn-to-rank.

1.4 METHODOGY

For the full development of this work, we split the research into the following parts:

1. **Literature review**: We cover the topics related to recommender systems reliability as well as techniques for collaborative filtering.

2. **Detect current literature's gaps**: Most research in recommender systems reliability is within collaborative filtering using a model-based approach. However, the methods do not demonstrate a generalization across different model architectures. The literature still leaves room for improvement in terms of design and confidence quality. Additionally, prior solutions for confidence in CF mainly focus on the rating prediction task, rather than a broad application for learn-to-rank.

3. **Choose datasets context**: As recommender systems can be applied to a wide range of different contexts, we choose datasets covering market and entertainment, as they are the most applied in the literature for the current problem.

4. **Confidence integration approach**: The literature comprises several different approaches, where we identified a trend towards distribution-based methods.

5. **Implement experimental evaluation setup**: Given the previous decision, it's possible to implement the employed models and confidence integration methods. Therefore, we need an algorithm to train and evaluate each method, including our proposal.

6. **Evaluation**: For the evaluation, we use common metrics for regression and ranking accuracy, along with metrics and visual analyses of the models' confidence outputs. Specifically, we assess two methods for integrating confidence: one for the rating prediction task and another for the pairwise learn-to-rank task.

This work is divided into two parts. In the first part, we perform an experimental evaluation of the four prior methods, OrdRec, CPMF, CBPMF, and LBD. Then, we propose confidence integration for GAT-based models, closing the first part, that is focusing on the rating prediction task. As a second part, focusing on the pair-wise learn-to-rank task, we introduce uncertainty-aware learning-to-rank, a method that integrates confidence estimation directly into the pairwise ranking framework. A complete glimpse of the work structure is as follows.

1.5 WORK STRUCTURE

The remainder of this work is as follows: Chapter 2 reviews recommender systems, covering recommendation techniques, collaborative filtering methods such as matrix factorization and graph attention networks, learning-to-rank approaches (pointwise, pairwise, and listwise), and evaluation methods. Chapter 4 presents distribution-based methods for confidence estimation in collaborative filtering models, including OrdRec, Confidence-Aware Matrix Factorization, and Learned Beta Distributions. Chapter 3 focuses on machine learning, discussing embedding-based models, model optimization, convergence, bias and variance, regularization methods, and solution modeling. Chapter 5 details related works, including taxonomy, independent confidence estimation, confidence calibration, general neural network approaches, distribution-based confidence modeling in recommender systems, alternative approaches leveraging confidence, synthesis, and research gaps. Chapter 6 presents the proposal, introducing probabilistic recommender graph

attention networks (PRGAT), Deep Graph Attention Networks model, and uncertainty-aware learn-to-rank (UALR). Chapter 7 presents the experimental evaluation of the first proposal, focusing on rating prediction. It outlines the methodology, data preparation, evaluation metrics, results, discussion, and limitations. Chapter 8 depicts the experimental evaluation of the second proposal, UALR. Finally, Chapter 9 provides the conclusion, highlights the contributions, and brings future directions.

Chapter 2

This chapter will demonstrate the foundational concepts and methodologies that underpin modern recommender systems, with a particular focus on CF approaches. Particularly, we present learn-to-rank methods using machine learning models. These concepts will provide a foundational understanding necessary for grasping the underlying problem and the proposed solution.

RECOMMENDER SYSTEMS

With the development of technology and the widespread adoption of the web in the late 20th century, information became increasingly accessible to people. However, this scenario also brought some challenges related to information overload (ALJUKHADAR; DAOUST, 2012). This means that users are exposed to an excessive and diverse amount of information coming from various sources and services: social networks, e-commerce, blogs, advertisements, news sites, wikis, emails, short message services, and many other sources that we cannot ignore due to the impact this information can have on our daily lives (MARQUES, 2016).

The abundance of information has become an obstacle for users, making it difficult to find relevant and helpful information. In this somewhat chaotic scenario, numerous technologies have been developed to alleviate this burden for users, aiming to suggest only truly relevant and appropriate information. RecSys, in a simpler sense, provides ranked lists of items, aiming to predict which products or services are most suitable, that is, which ones best fit, based on the registered user's experiences (NIKOLAKOPOULOS et al., 2022).

RecSys is an important tools that help users find items of interest amid the vast information available. They are widely used in various application domains, such as the service industry, content consumption, e-commerce, and online classification platforms. For example, YouTube, Spotify, Amazon, and Mercado Livre are sites that benefit from RecSys to offer their users personalized suggestions of videos, music, and products, respectively. In this way, RecSys contributes to increasing user satisfaction (PIYADASA; SILVA; KASTHURIRATHNA, 2022).

RecSys aims to provide a reduced amount of information (items) that matches a specific user's needs. The goal of these systems is to find an ordered list of items $\mathcal{I}^u \subset I$,

which maximizes the satisfaction of the user $u \in U$, where this list is

$$\mathcal{I}^u = \underset{i \in I - I_{rated}}{\text{argsort}} f(u,i) \, \forall u \in U, \tag{2.1}$$

here, $f(u,i)$ is a function $U \times I \rightarrow R'$. This algorithm generates a ranked list of items not viewed by the user, ordered by decreasing satisfaction (BOBADILLA et al., 2013).

Since this function, $f(u,i)$, is very complex, several approaches have been introduced to find a far but considered good approximation. These solutions can be broadly classified as CBF, demographic filtering (DF), and CF. Content-based filtering accommodates all strategies and algorithms that explore similarities between items. Demographic filtering follows the same strategy but utilizes demographic data, including personal information. CF utilizes user interactions with items, leveraging similarities between users or items, whether isolated or combined. The system can even be hybrid by combining one or more of those strategies. This work focuses on CF.

2.1 COLLABORATIVE FILTERING

Based on the preferences of other users in the past, the system, in collaborative filtering, recommends items to the active user. The similarity of user preferences is calculated based on the equivalence of their interaction history. This technique is considered one of the most popular in RecSys and the most widely implemented one (KOREN; RENDLE; BELL, 2022).

The basic principle is that a user *U1*'s rating of a new item i tends to be similar to that of another user *U2* if *U1* and *U2* have given similar ratings to other items. Likewise, it is likely that *U1* will rate items i and j similarly if other similar users have provided comparable ratings for both items (KOREN; RENDLE; BELL, 2022).

The CF methods are generally split into model-based, memory-based or neighborhood-based, and hybrid-based approaches (BURKE, 2007). In neighborhood-based methods, user-item ratings stored in the system are directly used to predict ratings for new items. The most well-known approaches are user-based and item-based filtering (KOREN; RENDLE; BELL, 2022).

In user-based systems, a user's preference for an item i is evaluated based on the ratings other users give for the same item. These users are referred to as neighbors, as they exhibit similar rating patterns. The neighbors of user *U1* are typically users *Un*, whose ratings for the items rated by both *U1* and *U2*, i.e., I_{U1U2}, are most correlated with those of *U1*.

Figure 2.1 illustrates the user-based approach in a book recommendation scenario. In a digital library environment, *U1* has shown interest in genres 1, 2, and 3, while *U2* prefers books in genre 3. Meanwhile, *U3* has indicated a liking for both genre 1 and genre 3. Based on this data, the recommendation system determines that *U3* has preferences closer to those of *U1*, thus suggesting genre 2 and genre 1 books to *U3*, since *U1* also enjoys these genres. In summary, *U1* is considered a neighbor of *U3* in the reading preference space.

On the other hand, the item-based approach predicts a user *U1*'s rating for an item i based on *U1*'s ratings for items similar to i. In this reasoning, two items are considered

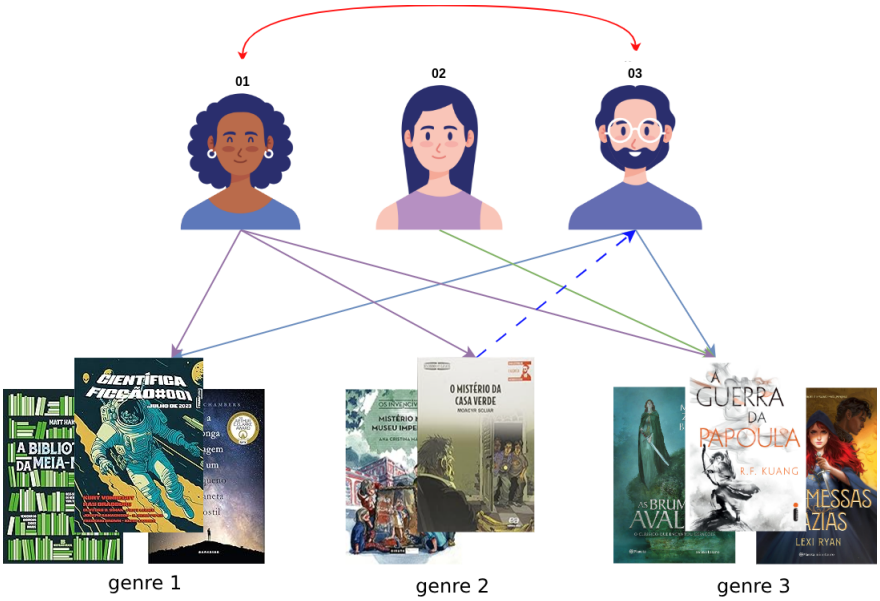

Figure 2.1: Illustration of user-based collaborative filtering in a recommender system. This figure represents a recommendation scenario where three users (01, 02, and 03) interact with books categorized into three genres (genre 1, genre 2, and genre 3). The connections between users and books indicate interactions, such as ratings or purchases. The red arrow between users 01 and 03 suggests a high similarity in their preferences, which forms the basis for user-based collaborative filtering. The continuous arrows represent interactions between the users and items. The dashed arrows represent a recommendation based on the interactions.

equivalent if many users in the system happen to rate these items similarly (KOREN; RENDLE; BELL, 2022).

An example of the item-based approach is illustrated in Figure 2.2 in the context of a book recommendation scenario. User 01 liked the books 04, 06, and 07; *U2* liked 04 and 06, and *U3* liked 06. Assuming the system needs to recommend a book to *U3*, this approach will search for books that received similar ratings to *06*. As a result, the system will recommend the book *04* to *U3* because this book received two ratings that *06* also received, meaning that *06* and *04* are considered neighbors.

Model-based approaches, in contrast, use these ratings to learn a predictive model, whereas neighborhood-based systems directly utilize stored ratings for prediction. The main idea is to reconfigure the user-item relationships with latent factors, thereby representing hidden characteristics of users and items in the system, such as the user's preference category and the item's category. This model is then trained using the available data and subsequently used to predict whether a user would or not like an unseen item (KOREN; RENDLE; BELL, 2022). This training process is performed using specific methods, such as pointwise or pairwise learn-to-rank.

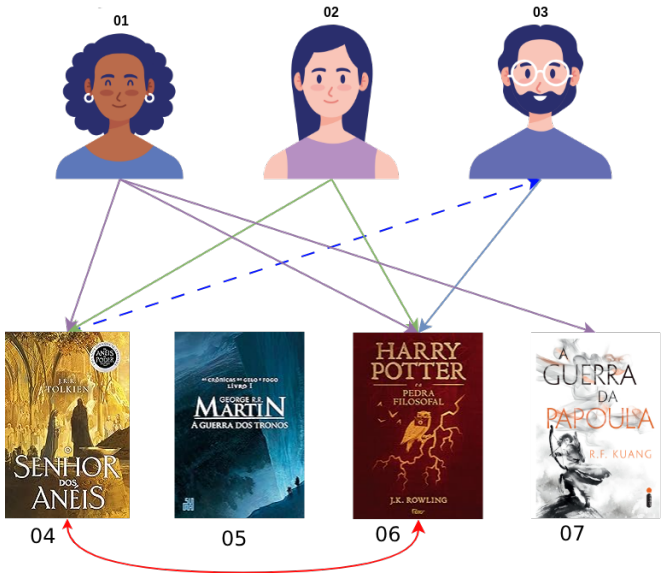

Figure 2.2: Illustration of item-based collaborative filtering in a recommender system. The figure represents a recommendation scenario where three users (01, 02, and 03) interact with different books (items 04, 05, 06, and 07). The connections between users and books indicate interactions such as ratings or purchases. The similarity between items is depicted with arrows, where a stronger relationship (red arrow) suggests a higher similarity between items based on user interactions. The continuous arrows represent interactions between the users and items. The dashed arrows represent a recommendation based on the interactions.

2.2 LEARNING TO RANK METHODS

Learning to rank refers to machine learning techniques that train models to rank items appropriately, particularly in recommender systems and information retrieval tasks. These methods are generally classified into three main categories: pointwise, pairwise, and listwise approaches. They are described in the next subsections.

2.2.1 Pointwise Approach

In the pointwise setting, the ranking task is framed as a regression or classification problem where each item is considered independently. The most common approach is rating prediction, where a regression is performed to estimate the item's relevance. The model learns to predict a relevance score $\hat{y}_{ui}$ for each item i and user u based on a ground-truth score y_{ui}. In the case of rating prediction, using a stochastic gradient descent-based algorithm, the objective is to minimize the mean squared error of any machine learning model, such as MF, (KOREN; BELL; VOLINSKY, 2009):

$$\mathcal{L}_{\text{pointwise}} = \sum_{(u,i)} (y_{ui} - \hat{y}_{ui})^2 \tag{2.2}$$

This approach is simple and interpretable, but it may fail to capture the relative order

between items. It also fails when, for example, explicit feedback from the users is missing.

2.2.2 Listwise Approach

Listwise methods consider the entire list of items for a user simultaneously, which optimizes the ranking quality at the list level rather than at the level of individual pairs. A classic example is ListNet (CAO et al., 2007), which defines a probability distribution over permutations of items based on their predicted scores $\hat{y}_{ui}$:

$$P(\pi|u) = \prod_{i=1}^{n} \frac{e^{\hat{y}_{u\pi(i)}}}{\sum_{k=i}^{n} e^{\hat{y}_{u\pi(k)}}} \tag{2.3}$$

where π denotes a permutation of items.

The ListNet loss minimizes the cross-entropy between the predicted and ground-truth permutation distributions, given any definition for the probability distribution:

$$\mathcal{L}_{\text{ListNet}} = -\sum_{u}\sum_{i} P^*_{ui} \log P_{ui} \tag{2.4}$$

where P^*_{ui} represents the target probability that item i should appear at a certain rank in the ground-truth list.

2.2.3 Pairwise Approach

Pairwise methods focus on relative preferences between item pairs. For example, BPR (RENDLE et al., 2012) assumes that for a user u, an observed item i should be ranked higher than an unobserved item j. The objective function is:

$$\mathcal{L}_{\text{BPR}} = -\sum_{(u,i,j)} \log \sigma(\hat{y}_{ui} - \hat{y}_{uj}) \tag{2.5}$$

where $\sigma(\cdot)$ is the sigmoid function. During training, a negative item j is sampled for each positive item i to enforce this preference constraint.

Pairwise methods like BPR and RankNet operate on pairs of items, while listwise methods like ListNet consider entire ranked lists. Although RankNet (BURGES et al., 2005) is sometimes described as "listwise-inspired", its loss is still pairwise:

$$\mathcal{L}_{\text{RankNet}} = -\sum_{(i,j)} [\bar{P}_{ij} \log P_{ij} + (1 - \bar{P}_{ij}) \log(1 - P_{ij})], \quad P_{ij} = \sigma(\hat{y}_{ui} - \hat{y}_{uj}) \tag{2.6}$$

where $\bar{P}_{ij}$ is the ground-truth preference (1 if i is preferred over j, otherwise 0). Although these methods are referred to as pairwise learn-to-rank, they still optimize ranking metrics and are therefore also equivalent to listwise approaches.

Note that all these loss functions rely on user and item identifiers, u and i, respectively. As discussed earlier in the chapter on collaborative filtering, these interactions naturally form a network of relationships among users and items. For this reason, we adopt a graph-based representation as our standard data format. The next section provides a detailed description of this representation.

2.3 GRAPHS DATA STRUCTURE

Since all the RecSys datasets contain at least one type of user interaction with items or other users, we can model the data as graphs. Then, this graph contains edges ($\mathbf{\Gamma}$), vertices (V), and node features ($\mathbf{X}$). We use capital letters for sets (e.g., V), bold letters for vectors, matrices, or tensors, and lowercase letters for scalars.

The X data structure is the entity features and is defined as a matrix,

$$\mathbf{X} = \begin{bmatrix} f_{0,1} & f_{0,2} & \cdots & f_{0,m} \\ f_{1,1} & f_{1,2} & \cdots & f_{1,m} \\ \vdots & \vdots & \ddots & \vdots \\ f_{N,1} & f_{N,2} & \cdots & f_{N,m} \end{bmatrix}. \tag{2.7}$$

$\mathbf{X}$ is the feature matrix for the nodes. Each row corresponds to a node, indexed from 0 to $N-1$, where N is the total number of nodes. Each column $f_{i,j}$ represents the j-th feature of the i-th node. Thus, $\mathbf{X}$ has dimensions $N \times m$, where N is the number of nodes and m is the number of features per node. The term $\mathbf{\Gamma}$ is the edge list matrix defining pairs of connections, defined as

$$\mathbf{\Gamma} = \begin{bmatrix} s_1 & s_2 & \cdots & s_n \\ t_1 & t_2 & \cdots & t_n \end{bmatrix}, \tag{2.8}$$

where s_i is the source node for the i-th edge, and t_i is the target node for the i-th edge. Each column $\begin{bmatrix} s_i \\ t_i \end{bmatrix}$ represents a directed edge from s_i to t_i, s_i and t_i are values belonging to the set of indices of the matrix $\mathbf{X}$, then, each s_i and t_i has a row representing their features each.

For users, the features can include intrinsic attributes, such as age, gender, location, work area, and interests, as well as statistical descriptors derived from their interactions, including user mean and variance, or group-level statistics. Similarly, item features may include intrinsic characteristics (e.g., genre, price, ratings) and interaction-based descriptors (e.g., item mean, variance, and group-level features). These features can be incorporated together with learned embeddings typically achieve better performance. An embedding is a vector in the latent space, defined as a lookup table where each entity has a learnable scalar, vector, or matrix—typically a vector (HAMILTON; YING; LESKOVEC, 2017). For example, given a position (such as an item or user ID), we obtain a vector. This vector is updated iteratively using the database.

Some databases natively provide graph-structured data, such as MovieLens, which offers a data frame similar to $\mathbf{\Gamma}$ but transposed and with an additional column, 'rating.' This rating represents the edge weight in the graph. This work utilizes machine learning models to predict the edge weight (rating) based on an edge element. One of the most common methods for performing this task is matrix factorization, which is presented in the next section.

2.4 MATRIX FACTORIZATION FOR RATING PREDICTION

MF is one of the most common methods, described as

$$\boldsymbol{R} \approx \boldsymbol{U} \times \boldsymbol{V}^T, \tag{2.9}$$

where $\boldsymbol{U}$ is the latent matrix representing the users embeddings and $\boldsymbol{V}$ is the latent matrix representing the items. This method aims to fit $\boldsymbol{U}$ and $\boldsymbol{V}$ using optimization algorithms such as Adam (GOODFELLOW; BENGIO; COURVILLE, 2016). $\boldsymbol{U}$ and $\boldsymbol{V}$ correspond to the learned embeddings of users and items, respectively, as previously defined in the context of graph-based representations. These embeddings capture latent features in a low-dimensional space, where each row in $\boldsymbol{U}$ represents a user embedding, and each row in $\boldsymbol{V}$ represents an item embedding. Since embeddings are typically optimized during training, $\boldsymbol{U}$ and $\boldsymbol{V}$ serve as entity representations that improve recommendation performance.

The literature presents other variations of MF, including singular value decomposition (SVD), alternating least squares (ALS), and non-negative MF (NMF) (KOREN, 2008; HIMABINDU; PADMANABHAN; PUJARI, 2018). In particular, SVD is one of the most famous variants of MF. The item rating, score, or relevance from a user for a given item is approximated by

$$r_{ui} = b_i + b_u + \mathbf{q}_i^T \left(\mathbf{p}_u + |R(u_{id})|^{-\frac{1}{2}} \sum_{j \in R(u_{id})} \mathbf{x}_j \right) \tag{2.10}$$

where b_i and b_u are learned scalar bias terms for item i and user u, respectively. $\mathbf{q}_i \in \mathbb{R}^D$ and $\mathbf{p}_u \in \mathbb{R}^D$ are latent vectors representing item i and user u in a D-dimensional latent space. These are typically implemented as embeddings. $\mathbf{x}_j \in \mathbb{R}^D$ is another latent vector associated with item j, used to model implicit feedback, that is, items the user has interacted with, regardless of explicit rating. $R(u_{id})$ denotes the set of items that user u_{id} has interacted with. The term $|R(u_{id})|^{-\frac{1}{2}} \sum_{j \in R(u_{id})} \mathbf{x}_j$ computes the average implicit feedback vector from the user's interactions, normalized by the number of interacted items. This implicit feedback vector is added to the explicit user latent vector $\mathbf{p}_u$, enriching the user representation before computing the dot product with $\mathbf{q}_i$. This model enhances the standard user-item interaction by incorporating implicit user behavior to improve recommendation accuracy, particularly when explicit ratings are sparse. Another model type is graph neural networks, which are applied to RecSys to leverage the intrinsic graph nature of the most common datasets for RecSys.

2.5 GRAPH ATTENTION NETWORKS

One particular implementation of GNN is the GAT architecture, which is powerful for processing graph-structured data. This architecture is well-suited for tasks that involve leveraging connections between entities, such as users and items in recommendation systems. Its advantage lies in its ability to incorporate structural information and feature attributes into a unified model, which provides a high representation of the input in a new latent space.

This way, it is possible to process graphs capturing user-user and item-item relationships and interactions. The model is fed with two data structures, $\mathbf{X}$ and $\mathbf{\Gamma}$.

GAT utilizes filters and attention mechanisms to weigh the importance of the connections and features. It dynamically learns the relationships and attributes that are most relevant to the task at hand. For example, given an edge list where the source nodes are users, and the target nodes are items, the node's features are projected into new feature spaces where complex relationships are represented (VELIčKOVIć et al., 2018). Specifically, consider

$$\text{inputs:} \quad \mathbf{X} \in \mathbb{R}^{N \times F}, \quad \mathbf{\Gamma} \in \mathbb{R}^{2 \times M}, \tag{2.11}$$

which is then applied to a graph attention layer. This layer results in

$$H_i = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W}\mathbf{X_j}\right), \tag{2.12}$$

where

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top[\mathbf{W}\mathbf{X_i} \parallel \mathbf{W}\mathbf{X_j}]\right)\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top[\mathbf{W}\mathbf{X_i} \parallel \mathbf{W}\mathbf{X_k}]\right)\right)}, \tag{2.13}$$

$$\mathbf{H} \in \mathbb{R}^{N \times F'}.$$

afterward, the concatenation aggregation is performed, where

$$\mathbf{Z} = \mathbf{H}[\mathbf{\Gamma}[0,:]] \parallel \mathbf{H}[\mathbf{\Gamma}[1,:]]$$

and

$$\mathbf{Z} \in \mathbb{R}^{M \times (2 * F')}.$$

The term $\mathbf{a}$ in Equation 2.13 is a feedforward NN, and $\mathbf{W} \in R^{F' \times F'}$ is also a learned matrix. The term α_{ij} represents the relevance of node j's features in constructing the final representation of node i. This representation is determined by the edges provided to the model, where i is the source node and j is the target node. Ultimately, the input feature matrix $\mathbf{X}$, representing users and items, is transformed into a new latent space, yielding the learned representations $\mathbf{Z}$. This means that each node's representation in $\mathbf{X}$ is updated through the GAT model and stored in $\mathbf{Z}$. This architecture can be incorporated into any neural network–based model, including those used for rating prediction. Depending on the method used for optimizing this model or matrix factorization, we evaluate the model differently. For rating prediction, regression metrics are often employed, such as the mean squared error or the mean absolute error. On the other hand, when evaluating rank quality, mean average precision and normalized discounted cumulative gain are often employed. These metrics are better described in the next section.

2.6 EVALUATION OF COLLABORATIVE FILTERING MODELS

This work employs standard MF and GNN models. For evaluating these models, error and ranking metrics such as RMSE, mean absolute error (MAE), and normalized discounted cumulative gain (NDCG) can be used. These metrics are based on the predicted rating

of the items for the target user. RMSE and MAE are error metrics. In particular, the RMSE calculates the root of the squared mean of error as

$$\text{RMSE} = \sqrt{MSE}\,. \tag{2.14}$$

The NDCG metric measures the model's ranking performance. Given a sorted list of items to be recommended to a specific user, NDCG measures how close this list is to the observed and expected lists. When we compute the model score for the top k elements, we can refer to the NDCG score as NDCG@k. Specifically, the score for top k is

$$\text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}\,, \tag{2.15}$$

where

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)}\,, \tag{2.16}$$

$$\text{IDCG@}k = \sum_{i=1}^{k} \frac{\text{rel}_i^*}{\log_2(i+1)}\,, \tag{2.17}$$

and rel_i is the relevance score of the item at rank i. However, the Ideal Discounted Cumulative Gain (IDCG@k) uses the list sorted by the true relevance, whilst DCG@k uses the list sorted by the predicted relevance.

In addition to NDCG, other ranking-based metrics such as Precision and MAP can also be used. Precision@k measures the proportion of recommended items in the top k positions that are truly relevant:

$$\text{Precision@}k = \frac{1}{k}\sum_{i=1}^{k} \mathbb{I}(\text{rel}_i = 1),\,, \tag{2.18}$$

where $\mathbb{I}(\cdot)$ is the indicator function returning 1 if the item is relevant, and 0 otherwise.

MAP, on the other hand, evaluates the quality of the ranking by averaging the precision at each position where a relevant item occurs, across all relevant items and users:

$$\text{MAP} = \frac{1}{|U|}\sum_{u\in U}\left(\frac{1}{|R_u|}\sum_{k\in R_u}\text{Precision@}k\right)\,, \tag{2.19}$$

where R_u is the set of relevant items for user u. MAP emphasizes the importance of retrieving relevant items early in the ranked list. While metrics like MAP and Precision focus directly on binary relevance, NDCG incorporates graded relevance and positional discounting.

The other metric used in this dissertation is the Pearson correlation, which measures the relationship between the model's confidence and the error. The Pearson correlation coefficient is a statistical measure that quantifies the strength and direction of a linear relationship between two variables. Formally, given paired samples ((x_k, y_k)) for (k = 1, . . . , n), the Pearson correlation (r) is defined as (RODGERS; NICEWANDER, 1988)

$$r = \frac{\sum_{k=1}^{n}(x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{n}(x_k - \bar{x})^2}; \sqrt{\sum_{k=1}^{n}(y_k - \bar{y})^2}}, \tag{2.20}$$

where $(\bar{x})$ and $(\bar{y})$ are the sample means of (x) and (y), respectively. The coefficient ranges from –1 to 1, with values near 1 indicating a strong positive linear relationship, values near –1 indicating a strong negative relationship, and values near 0 indicating weak or no linear association. In the context of confidence estimation, Pearson correlation is particularly useful for assessing how a model's confidence relates to its prediction error. A well-calibrated confidence estimator should exhibit a strong negative correlation between confidence and error, meaning that higher confidence should correspond to lower prediction error.

2.7 SUMMARY

In conclusion, this chapter has shown the foundational concepts and methodologies underpinning modern recommender systems, particularly CF approaches. The discussion highlighted the evolution from traditional filtering strategies to sophisticated model-based techniques such as MF and GAT to address the core objective of maximizing user satisfaction through personalized item recommendations. The chapter further underscores the importance of robust evaluation frameworks, utilizing error metrics like RMSE alongside ranking-focused measures such as NDCG@k and MAP to assess both predictive accuracy and the quality of recommendation rankings. As the focus of this dissertation is on confidence of collaborative filtering models, we also outlined the Pearson correlation as a confidence-error relationship measurement. These concepts provide a foundation for understanding the underlying problem and the proposed solution. The next chapter presents more details about confidence in collaborative filtering models for recommender systems.

Chapter

3

This chapter will present the fundamental machine learning concepts essential for understanding the proposed solution. The techniques to be discussed will encompass optimisation methods, embedding strategies, normalisation processes, goal functions, performance analysis frameworks, activation functions, and various modelling approaches.

MACHINE LEARNING TECHNIQUES FOR COLLABORATIVE FILTERING

Artificial Intelligence is a field of Computer Science that aims to solve problems that are difficult to address using a sequence of formal logical steps. These are problems for which it is challenging or impossible to write explicit algorithms, so AI techniques are employed as an alternative. Machine learning (ML) is an approach that seeks to automatically produce models that capture knowledge. Deep Learning is an ML approach that aims to capture the best input data representation and the function that maps the relationships between input data and target (GOODFELLOW; BENGIO; COURVILLE, 2016, Chapter 5). In recommendation systems, modeling the relationship between users and items is a complex task. Specifically, a model-based approach in collaborative filtering utilizes machine learning techniques to design, train, and evaluate these models. Matrix factorization and graph neural network, the most common among collaborative filtering models, can both be treated as machine learning models based on embeddings.

3.1 EMBEDDING-BASED MODEL

Embedding-based models learn representations from the input in the latent space; one example is the NN. NN is a versatile ML model that can be used for approximate functions (MITCHELL, 1997, chapter 4). A NN can be linear or non-linear, stacked in sequential layers or parallel. We refer to it as a Deep Neural Network because it has depth when stacked sequentially. For example, a deep matrix factorization can be obtained through applying neural network layers over the product of the user and item embeddings. We can make a model that can approximate non-linear functions by adding a nonlinear activation function over a layer output, such as ReLU and Softmax (GOODFELLOW; BENGIO; COURVILLE, 2016, Chapter 6).

This work employs DL techniques for developing embedding-based models that approximate the relationship between users and items given a set of iterations in different public datasets. These models must be optimized to ensure that they fit the desired function.

3.2 MODEL OPTIMIZATION

We have supervised, semi-supervised, and unsupervised methods for optimising a model. This work focuses on supervised learning, where it is necessary to obtain a set of paired samples (x, y), where x represents the input data and y represents the label data. In our case of employing graph data for collaborative filtering, a x sample is an edge connection, which is a tuple with user and item IDs, (u, i). The y definition depends on the loss function defined for the model. We use the Stochastic Gradient Descent (**SGD**) algorithm to optimise the model's loss. This algorithm performs forward and backward propagation stages at each step during optimisation time (HAYKIN, 2008, chapter 4). The forward pass consists of performing the mathematical operations of the model by providing the input values, x, and obtaining the model's output inference, $\hat{y}$. For example, consider a sample (x,y) = ([0,0],1) applied to a simple NN with only one layer and only one neuron; the feedforward results in (MITCHELL, 1997, chapter 4)

$$f(\boldsymbol{x}; \boldsymbol{\Theta}) = f(\boldsymbol{x}; \boldsymbol{W}, b) = \boldsymbol{x}^T \boldsymbol{W} + b = 0 \times 0.5 + 0 \times (-0.7) + 0.3 = 0.3\,. \tag{3.1}$$

In this model, Equation 3.1, the input ([0,0]) is passed directly to the fully connected neural network layer, composed of the weights ($\mathbf{W} = [0.5, -0.7]$) and bias ($b = 0.3$). When using embeddings, an index embedding retrieval is often applied using the users' and items' IDs instead. During the backward stage, it calculates the gradient of each layer based on the output error and updates the parameter values, such as

$$\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \epsilon \nabla_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta}, \boldsymbol{x}, \boldsymbol{y})\,, \tag{3.2}$$

where J is the objective function and ϵ is the learning rate.

The cost function can be expressed by the expected value ($\mathbb{E}$) of the loss function over the dataset ($(x, y) \sim p_{\text{data}}$), such as $J(\Theta) = \mathbb{E}_{(x,y)\sim p_{\text{data}}} L(f(x; \Theta), y)$. A loss function should be defined as the starting point of problem-solving modeling. Regression and classification are commonly performed for most RecSys problems solved with embedding-based models. For regression tasks, minimising the mean squared error (MSE) (Equation 3.3) or BPR is commonly defined as a goal. In the Equation 3.3, n is the number of samples, (GOODFELLOW; BENGIO; COURVILLE, 2016, Chapter 8).

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2\,. \tag{3.3}$$

Still considering the example with sample ([0,0],1), considering that regression has chosen with $\frac{1}{2}MSE$ as loss function, the loss is calculated as

$$\text{Loss} = \frac{1}{2}(f(x; \Theta) - y)^2 = \frac{1}{2}(0.3 - 1)^2 = \frac{1}{2} \times (-0.7)^2 = 0.245\,. \tag{3.4}$$

The backpropagation updates the model's parameters, $\theta = W_1, W_2, b$, using the gradients of the loss function. Therefore, the gradient is applied to all weights and biases (b, not *Bias*), as expressed in equations 3.5, 3.6, and 3.7. These gradients represent the $\nabla_\Theta J(\Theta, x, y)$ term of the Equation 3.2, where $\mathbf{W}$ is updated concerning the variation of the loss in terms of W_1 and W_2, the same for b. Thus, in this example, W_1 and W_2 remain the same as in the previous case because the gradient for that sample is zero. The bias is now updated to $b \leftarrow b - 0.7$.

$$\frac{\partial \text{Loss}}{\partial f} = f(x; \Theta) - y = 0.3 - 1 = -0.7\,. \tag{3.5}$$

$$\frac{\partial \text{Loss}}{\partial W} = \frac{\partial \text{Loss}}{\partial f} \times \frac{\partial f}{\partial W} = -0.7 \times x = -0.7 \times [0, 0] = [0, 0]\,. \tag{3.6}$$

$$\frac{\partial \text{Loss}}{\partial b} = \frac{\partial \text{Loss}}{\partial f} \times \frac{\partial f}{\partial b} = -0.7 \times 1 = -0.7\,. \tag{3.7}$$

Due to the nature of most real-world datasets, which often exhibit high sparsity, noise, and non-convexity, the presented SGD faces several convergence issues. The gradients exhibit stochastic behavior along the samples, so it has been necessary to consider more than just the gradient and a single sample to define the direction and step size for parameter updates. Otherwise, the model might not converge or present a high training time. The next section outlines the most common issues and provides guidance on how to resolve them.

3.3 MODEL CONVERGENCE

SGD uses learning rate decay to control the step size for faster convergence. But it still has the stochastic nature of the gradient. The average of the minibatch gradient is used as an approximation of the gradient to reduce this noise. One minibatch, $\mathcal{B}$, is a set of samples randomly taken from the dataset. The expression of the parameter update is changed as follows:

$$\mathbf{\Theta} \leftarrow \mathbf{\Theta} - \alpha \frac{1}{|\mathcal{B}|} \sum_{(x_i, y_i) \in \mathcal{B}} \nabla_\Theta J(\mathbf{\Theta}, x_i, y_i)\,. \tag{3.8}$$

While SGD can be slow and unstable during convergence, techniques such as gradient clipping, weight initialisation, momentum, adaptive learning rate, and model normalisation are used to address these issues. Gradient clipping ensures that the gradients do not exceed a certain threshold. When initialising the weights, at least, it seeks to avoid symmetry among the neurons. We have Xavier Weight Initialisation (Equation 3.9), Normalised Xavier Weight Initialisation (Equation 3.10), and Weight Initialisation for ReLU (He), shown in Equation 3.11. In those equations, 3.9, 3.10, and 3.11, m is the number of inputs to the neuron, k is the number of outputs, W is the weight matrix, U is the Uniform distribution, and G is the Gaussian distribution. *He initialisation* was introduced to better initialise for NN with ReLU activation functions (GOODFELLOW; BENGIO; COURVILLE, 2016, Chapter 8).

$$\boldsymbol{W} = U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]. \tag{3.9}$$

$$\boldsymbol{W} = U\left[-\frac{\sqrt{6}}{\sqrt{n+k}}, \frac{\sqrt{6}}{\sqrt{n+k}}\right]. \tag{3.10}$$

$$\boldsymbol{W} = G(0.0, \sqrt{\frac{2}{n}}). \tag{3.11}$$

In physics, momentum is defined as mass times velocity, representing the quantity of an object's motion, including its direction and magnitude, as $p_o = m_o \cdot v_o$, where m_o is the mass and v_o is the velocity of the object. In the context of ML optimisation, momentum refers to the propagation of the previous gradient's direction and magnitude, accounting for the accumulation of past gradients, as expressed in 3.12. The idea of an adaptive learning rate is to increase the magnitude of the update when the gradient direction is consistent across steps and decrease it when the direction changes. In this Equation, 3.12 α is the level of effect of momentum, and $m = |\mathcal{B}|$. The parameter update is defined as $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} + \boldsymbol{v}$.

$$\boldsymbol{v} \leftarrow \alpha \cdot \boldsymbol{v} - \epsilon \cdot \nabla_{\Theta}\left(\frac{1}{m}\sum_{i=1}^{m} L\left(f\left(x^{(i)}, \boldsymbol{\Theta}\right), y^{(i)}\right)\right). \tag{3.12}$$

Algorithms based on SGD, such as AdaGrad, RMSProp, and Adam, are presented to consider adaptive learning rates. AdaGrad scales the learning rate by dividing it by the sum of the historical gradients. The RMSProp performs this scaling using an exponentially decaying gradient average, giving recent gradients more effect. The Adam algorithm combines the exponential learning rate decay with momentum.

Other techniques for stabilizing the training process and regularizing the model in some way include Batch Normalization and Layer Normalization. The embedding process is discussed in section 3.5. Batch Normalisation (Equation 3.15) normalises the layer's output separately to follow a standard Normal Distribution across a minibatch, while Layer Normalisation (Equation 3.18) normalises across its layer outputs with a single sample. These normalisation techniques stabilise the training process, especially in situations where the distribution of inputs changes (covariate shift).

$$\boldsymbol{\mu_b} = \frac{1}{m}\sum \boldsymbol{H_i}. \tag{3.13}$$

$$\boldsymbol{\sigma_b} = \sqrt{\delta + \frac{1}{m}\sum(\boldsymbol{H_i} - \boldsymbol{\mu})^2}. \tag{3.14}$$

$$\boldsymbol{H}'_b = \frac{\boldsymbol{H} - \boldsymbol{\mu_b}}{\boldsymbol{\sigma_b}}. \tag{3.15}$$

Equations 3.13 and 3.14 represent the mean and variation of the minibatch, respectively, where δ is a small number for 0 avoidance. In Equations 3.15 and 3.18, $H_{<>}$

and $H'_{<>}$ are the layer outputs and next normalized outputs, respectively. The collected averages of these values from training are used in inference time. In Equation 3.18, β and ϵ_l are optimizible parameters. In equations in (3.16) and (3.17), D is the output of a layer.

$$\mu = \frac{1}{|D|} \sum_{D_i \in D} D_i \,. \tag{3.16}$$

$$\sigma^2 = \frac{1}{|D|} \sum_{D_i \in D} (D_i - \mu)^2 \,. \tag{3.17}$$

$$\boldsymbol{H}'_l = \gamma \frac{\boldsymbol{H}_{-}\mu}{\sqrt{\sigma_l^2 + \epsilon_l}} + \beta \,. \tag{3.18}$$

By stabilizing the activations and preventing extreme fluctuations, such as those arising from perturbations in the previous layer or from exploding values, this normalization step enhances the model's robustness during training. It also serves as an implicit regularizer by constraining the distribution of intermediate representations, which helps reduce the model's overall Variance and mitigates overfitting. In the next section, we introduce the concepts of Bias and Variance in detail, and afterward discuss how the Variance of the model can be reduced.

3.4 BIAS AND VARIANCE

Measuring the performance and quality of a model during experiments is essential. Two critical aspects that must be analysed throughout the training process are Bias and variance. Bias measures how far the model parameters are from the optimal parameter values; the goal is then to minimize them (SMOLA; VISHWANATHAN, 2010, chapter 2). The Bias in a train step n can be expressed as defined by the Equation 3.19. The Equation 3.19, $\hat{\theta}_n$ is the true underlying values of parameters, $\hat{\theta}_n$ is the current model parameters, and $\mathbb{E}[\hat{\theta}_n]$ is the expected value of $\hat{\theta}_n$.

$$\text{Bias}(\hat{\boldsymbol{\theta}}_{\boldsymbol{n}}) = \mathbb{E}[\hat{\boldsymbol{\theta}}_{\boldsymbol{n}}] - \boldsymbol{\theta} \,. \tag{3.19}$$

The model can reach the optimal parameter values for the fit dataset when it is asymptotically unbiased, when

$$\lim_{n \to \infty} \mathbb{E}(\hat{\boldsymbol{\theta}}) = \boldsymbol{\theta} \,. \tag{3.20}$$

The variance measures how the estimator would vary as the dataset is changed. For example, the model's sensitivity can be measured by running it on the validation and test split data, where the validation and test splits represent a variation of the train dataset. Generalization error refers to the error of the model when applied to unseen data. In many applications, this error is obtained by splitting the data into train, validation, and test datasets, where the validation dataset and test data are not used for training the model. The generalisation error can be estimated using the test dataset. The validation dataset provides an indication of how the model will perform on unseen data. Figure 3.1 shows that Bias and Variance are components of Generalisation error (GOODFELLOW;

BENGIO; COURVILLE, 2016, Chapter 5). A good strategy for obtaining a model with low bias is to augment its capacity, but this tends to increase high variance. Regulation methods can be used to decrease the model's variance.

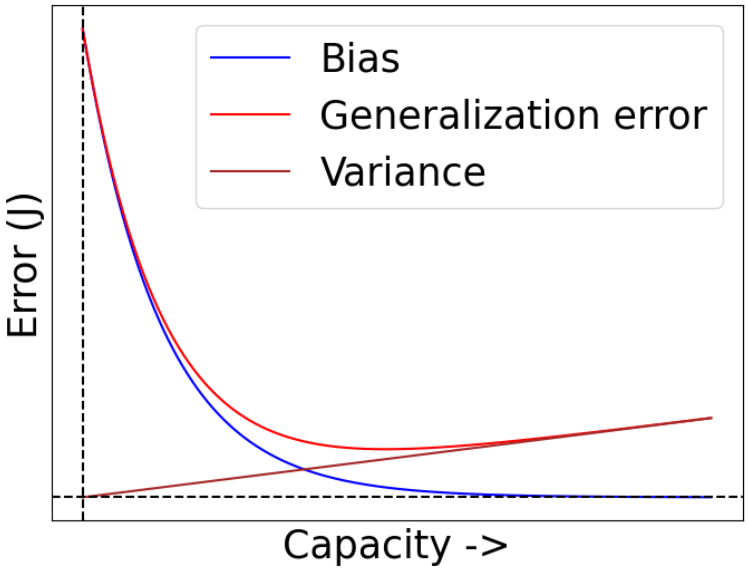

Figure 3.1: Illustration of the relationship between Generalisation error with Bias and Variance of the model. The exponential decay form represents the Bias, and the straight-line form represents the Variance.

3.5 REGULARIZATION METHODS

Among regularization techniques, there are *L2* parameter embedding, *L1* parameter regularization, dropout, and early stopping. *L1* and *L2* normalisation are usually applied only to the weights of the NN or embeddings. They are called parameter regularization, which adds a penalty to the loss function, J. Then, the loss expression is modified to the Equation 3.21. In this equation, α is the weight for the penalty term, $\Omega(\boldsymbol{\Theta})$. The *L2* regularization, expressed by Equation 3.22, drives the weights closer to the origin, where the weight updating is performed as defined in the expression in 3.23, where ϵ is the learning rate.

$$\hat{J}(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta}) \,. \tag{3.21}$$

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{W}\|_2^2 = \frac{1}{2}\left(w_1^2 + w_2^2 + \cdots + w_n^2\right) \,. \tag{3.22}$$

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \epsilon\left(\alpha\Omega + \nabla_w J(\boldsymbol{W}; \boldsymbol{X}, \boldsymbol{y})\right) \,. \tag{3.23}$$

On the other hand, *L1* regularization, as described by Equation 3.24, can be used for feature selection because it results in a more sparse solution, thereby setting zero for features that can be discarded. The loss function is modified as Equation 3.25 describes. The

early stopping strategy saves the best version of the model in terms of a metric, usually validation error, during the training time and stops fitting when the algorithm reaches a predefined number of epochs without any improvement (GOODFELLOW; BENGIO; COURVILLE, 2016, Chapter 7).

$$\Omega(\boldsymbol{\theta}) = \|\boldsymbol{W}\|_1 = \sum_1 |w_i| \quad . \tag{3.24}$$

$$\hat{J}(\boldsymbol{W}; \boldsymbol{x}, \boldsymbol{y}) = \alpha\|\boldsymbol{W}\|_1 + J(\boldsymbol{W}; \boldsymbol{x}, \boldsymbol{y}) \, . \tag{3.25}$$

The dropout technique can be understood as a method of assembling different NN models and sharing parameters by generating submodels from the original. This is performed by randomly dropping a unit's percentage of the network during training time. This random dropping is done by setting zero for the randomly chosen values. Therefore, the weights of the subnetworks are updated in each batch, then the whole network (GOODFELLOW; BENGIO; COURVILLE, 2016, Chapter 6). Figure 3.2 shows an example of an NN model with one hidden layer, illustrating that the dropout application results in different subnets. The hidden layer is called hidden because the expected output values are not explicitly defined.

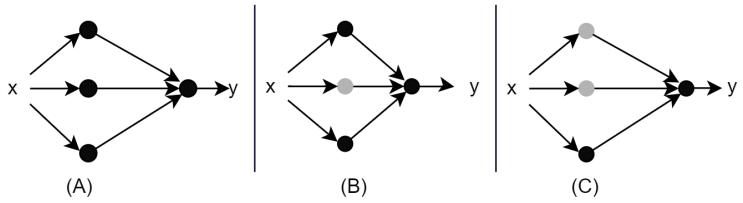

Figure 3.2: A NN with two layers: one hidden layer and one output layer. (A) No neurons are dropped. (B) One neuron (gray) is dropped. (C) Two neurons are dropped. Note that (A), (B), and (C) result in different NNs.

All of those techniques improve the model's optimisation process. However, other aspects should be considered, such as the model size and the parameterisation of those techniques. There is still no optimal methodology for defining the model architecture in the DL approach. Hence, this modelling must be done through experiments, mostly based on experience and heuristics.

3.6 SUMMARY

This chapter presented the fundamental machine learning concepts essential for understanding the proposed solution. The discussed techniques encompass optimization methods, embedding strategies, normalization processes, goal functions, performance analysis frameworks, activation functions, and various modeling approaches. The next chapter presents the literature on confidence estimation or integration in models for collaborative filtering in recommender systems.

Chapter

4

This chapter will present the most common methods for confidence integration in collaborative filtering models for recommendation systems. Specifically, we focus on distribution-based methods.

CONFIDENCE IN RECOMMENDER SYSTEMS

The term confidence in recommender systems carries different interpretations. This dissertation treats confidence as a measure of the model's own certainty, regardless of the user's actual preference (KOREN; SILL, 2011; WANG et al., 2018a; KNYAZEV; OOSTERHUIS, 2023). Under this second interpretation, given a model's prediction, we also expect it to provide a value representing how certain it is that the prediction is correct. If the model is well calibrated, these confidence values should be inversely proportional to the model's error: when confidence is high, the prediction should be reliable, whereas low confidence should correspond to a higher likelihood of error. Uncertainty is also a term used in this context. However, it is inversely proportional to the confidence. Rather than measuring the uncertainty, this dissertation chooses to explore confidence in the models for collaborative filtering (COSCRATO; BRIDGE, 2023). Such confidence outputs are common in several neural network–based models, but for other tasks.

For instance, in classification tasks, sigmoid or softmax activation functions are typically applied as the final layer of a neural network that can be interpreted as a likelihood score for the class (GOODFELLOW; BENGIO; COURVILLE, 2016, Chapter 6). In binary classification, where sigmoid is often used, the model outputs the probability that an event will occur—a value that can also be interpreted as the model's certainty in its prediction. In multilabel classification, which also utilizes a sigmoid, the interpretation extends to multiple labels, each with its own associated probability. In multiclass classification, where softmax is applied, the model outputs a probability distribution over mutually exclusive classes. However, these strategies are not commonly employed in recommendation tasks (KOREN; BELL; VOLINSKY, 2009; KOREN; SILL, 2011; WANG et al., 2018a; KNYAZEV; OOSTERHUIS, 2023).

As discussed in Section 2.2, collaborative recommendation system approaches can be categorized into pointwise or rankwise approaches. In pointwise settings, the goal is to predict the score that a user would assign to an item. Rating prediction is a common example, where collaborative filtering models perform a regression task rather than

a classification task. Prior work has shown that classification is not suitable for predicting user ratings because rating scales impose an ordinal structure (e.g., $1 < ... < 5$) (COSCRATO; BRIDGE, 2023; KNYAZEV; OOSTERHUIS, 2023; GOHARI; ALIEE; HAGHIGHI, 2018; WANG et al., 2018a; KOREN; SILL, 2011). Consequently, rating prediction models do not produce a probability representing the likelihood that the predicted rating is correct, nor any other explicit measure of confidence. In rankwise approaches, the objective is to predict a score that enables sorting items by relevance for a user. As in rating prediction, this also constitutes a regression task, and the predicted relevance score is not accompanied by an indication of whether the score is reliable or not (ZHANG; GUO; CHEN, 2016; WANG; KADIOĞLU, 2023; COSCRATO; BRIDGE, 2023).

The literature includes several methods for integrating confidence into collaborative filtering models, as discussed in the introduction to this dissertation. Among these, distribution-based approaches stand out as theoretically grounded: they allow confidence to emerge directly from the model's training process, naturally tied to the model's outputs without introducing additional complexity. This characteristic is particularly valuable, as it enables confidence to reflect the model's own uncertainty in a unified learning framework rather than relying on post-processing steps. These methods will be compared with one another in the related works chapter (Chapter 5). Therefore, as we focus on distribution-based methods, the following section brings more details.

4.1 CONCEPTS

From statistics, when we infer a value over a population, such as the mean, we also bring a confidence interval. However, this inference is based on certain assumptions, such as the population following a specific distribution (e.g., a Normal distribution). For instance, let $(X_1, X_2, \ldots, X_n)$ be a random sample from a normal distribution with mean (μ) and variance (σ^2). A confidence interval for a parameter (θ) is a random interval $[L(X), U(X)]$ such that (HOGG; CRAIG, 1965, Chapter 4.2) (WATT; MCCLEERY; HART, 2007, Chapter 2)

$$P\left(L(X) \leq \theta \leq U(X)\right) = 1 - \alpha, \tag{4.1}$$

for a chosen significance level α(typically 0.05), where L(X) is the lower and U(X) is the upper bound of this interval. The probability that the interval $[L(X), U(X)]$ contains the true parameter θ, over repeated samples, is $1 - \alpha$. Under this assumption, the sampling distribution of the sample mean is

$$\bar{X} \sim N\left(\mu,\ \frac{\sigma^2}{n}\right). \tag{4.2}$$

Because the distribution of $\bar{X}$ depends on the unknown parameters μ and σ, we standardize it to form a pivotal quantity whose distribution is known and does not depend on any unknown parameters. This is done by centering at μ and scaling by the standard error $\sigma/\sqrt{n}$ (HOGG; CRAIG, 1965)[Chapter 3.4, 4.2]:

$$Z = \frac{\bar{X} - \mu}{\sigma/\sqrt{n}}. \tag{4.3}$$

This transformation yields a standard normal random variable,

$$Z \sim N(0, 1). \tag{4.4}$$

Since the distribution of (Z) is known, we can use the fact that, for any $\alpha \in (0, 1)$,

$$P\left(-z_{\alpha/2} \leq Z \leq z_{\alpha/2}\right) = 1 - \alpha, \tag{4.5}$$

where $z_{\alpha/2}$ is the upper $\alpha/2$ quantile of the standard normal distribution. Substituting the expression for Z into the probability statement gives

$$P\left(-z_{\alpha/2} \leq \frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \leq z_{\alpha/2}\right) = 1 - \alpha. \tag{4.6}$$

We then solve this inequality for the unknown parameter μ. Multiplying all terms by $\sigma/\sqrt{n}$ and rearranging yields

$$\bar{X} - z_{\alpha/2}\frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{X} + z_{\alpha/2}\frac{\sigma}{\sqrt{n}}. \tag{4.7}$$

Thus, the $100(1 - \alpha)\%$ confidence interval for the population mean μ is

$$\bar{X} \pm z_{\alpha/2}\frac{\sigma}{\sqrt{n}} \tag{4.8}$$

This interval provides a range of plausible values for μ, based on the observed sample mean and the assumed normal distribution of the data. This means that the true mean, μ, would fall within this interval with $100 \cdot (1-\alpha)\%$ of confidence. This formulation serves as the basis for integrating confidence into collaborative filtering models, as seen in OrdRec (KOREN; SILL, 2011) and in this dissertation. From it, several possibilities emerge: the model (e.g., matrix factorization-based) may be designed to predict an interval for a given confidence level, to predict the confidence level associated with a given interval, or even to jointly predict both the interval and the confidence level. For instance, for rating prediction, we can make the model estimate the mean and standard deviation of the distribution. As a confidence estimation, we can also model the probability of observing a predicted rating, $P(r_{pred} = r|\Theta)$. In the next section, we cover prior distribution methods from the literature. Particularly, we explore four methods: OrdRec (KOREN; SILL, 2011) (Section 4.2), Confidence-aware Probabilistic Matrix Factorization (CPMF) and Confidence-aware Bayesian Probabilistic Matrix Factorization (CBPMF) (WANG et al., 2018a) in Section 4.3, and Learned Beta Distributions (LBD) (KNYAZEV; OOSTERHUIS, 2023) in Section 4.4.

4.2 CONFIDENCE INTEGRATION WITH ORDREC

OrdRec, as described by Koren and Sill (2011), is a collaborative filtering method that models user ratings as ordinal outcomes defined by thresholds rather than continuous regression. The approach assumes that for each user-item interaction, the latent preference score follows a Normal distribution, which is then mapped to discrete rating categories through threshold parameters. While the original implementation builds upon the SVD++ framework, the authors emphasize that the method can be generalized to any predictive model capable of producing real-valued scores y_{ui}. The core innovation lies in treating ratings as ordered categories (e.g., 1 to 5 stars) with user-specific interpretation scales.

The ordinal nature of ratings is captured through $S-1$ thresholds, where S is the number of possible rating levels (e.g., $S=5$ for 1–5 stars). The first threshold t_1^u is a learned scalar specific to each user u. Subsequent thresholds are defined recursively to enforce ordinality:

$$t_{r+1}^u = t_r^u + e^{\beta_r^u}, \quad r = 1, \ldots, S-2, \tag{4.9}$$

where β_r^u are user-specific parameters ensuring $t_1^u \leq t_2^u \leq \cdots \leq t_{S-1}^u$. The probability that a user's rating r_{ui} is less than or equal to r is modeled using the logistic function:

$$P(r_{ui} \leq r \mid \Theta) = \sigma(t_r^u - y_{ui}) = \frac{1}{1 + e^{y_{ui} - t_r^u}}, \tag{4.10}$$

where Θ represents all model parameters, and y_{ui} is the predicted score. The probability of observing exactly rating r is derived based on the difference between adjacent cumulative probabilities:

$$P(r_{ui} = r \mid \Theta) = P(r_{ui} \leq r \mid \Theta) - P(r_{ui} \leq r-1 \mid \Theta). \tag{4.11}$$

The model is trained by minimizing the negative log-likelihood of observed ratings:

$$\mathcal{L}(\Theta) = -\sum_{(u,i,r)\in\mathcal{R}} \log P(r_{ui} = r \mid \Theta), \tag{4.12}$$

optimized via a stochastic gradient descent-based algorithm.

To enhance ranking performance, OrdRec introduces a secondary calibration step. After fitting the ordinal model, a set of ranking weights $\mathbf{w} \in \mathbb{R}^S$ is learned to refine item ordering. For pairs of items (i, j) where the observed rating $r_{ui} > r_{uj}$, the difference in their predicted probability distributions $\Delta P_u(i,j) = P(r_{ui} = r \mid \Theta) - P(r_{uj} = r \mid \Theta)$ is computed. The weights $\mathbf{w}$ are then optimized to maximize:

$$F(\mathcal{T} \mid \mathbf{w}) = \sum_{(i,j)\in\mathcal{T}} \sigma(\mathbf{w}^T \Delta P_u(i,j)), \tag{4.13}$$

where $\mathcal{T}$ is the set of correctly ordered item pairs, and σ is the sigmoid function. This post-hoc calibration ensures that items with higher true ratings are ranked more prominently, while the original ordinal probabilities remain intact. The two-stage training process—first learning the rating distribution. Optimizing the ranking weights allows the model to maintain probabilistic interpretability while improving recommendation quality.

During inference, the model first computes the real-valued score y_{ui} for a user-item pair (u, i). This score is transformed into a probability distribution over all possible rating levels $r \in \{1, \ldots, S\}$ using the ordinal threshold model:

$$P(r_{ui} = r \mid \Theta) = \sigma(t_r^u - y_{ui}) - \sigma(t_{r-1}^u - y_{ui})\,. \tag{4.14}$$

The predicted rating $\hat{r}_{ui}$ is selected as the value r with the highest probability $P(r_{ui} = r \mid \Theta)$, which naturally serves as a confidence measure for that prediction.

For ranking tasks, where the goal is to sort items by user preference, the model utilizes the calibrated scores s_{ui}. These are derived by weighting the full probability distribution with the learned ranking weights $\mathbf{w}$:

$$s_{ui} = \mathbf{w}^T \mathbf{p}_{ui}, \quad \text{where} \quad \mathbf{p}_{ui} = \begin{bmatrix} P(r_{ui} = 1 \mid \Theta) \\ \vdots \\ P(r_{ui} = S \mid \Theta) \end{bmatrix}. \tag{4.15}$$

4.3 CONFIDENCE-AWARE MATRIX FACTORIZATION FOR RECOMMENDER SYSTEMS

This study presents two implementations of a confidence-aware matrix factorization model. We begin by describing the CPMF (WANG et al., 2018a) variant.

As recapped in Section 2.4, let $\mathbf{U}$ and $\mathbf{V}$ denote the user and item embeddings, respectively. Assuming the rating r_{ui} follows a normal distribution, it is modeled as:

$$r_{ui} \sim \mathcal{N}(\mu_{ui}, \nu^{-1})\,,$$

where the mean is given by:

$$\mu_{ui} = \mathbf{U}_i^\top \mathbf{V}_j\,,$$

and the variance is defined as:

$$\nu = \frac{1}{\gamma_{U_i} \cdot \gamma_{V_j} \cdot \alpha_p}\,,$$

in which γ_{Ui} and γ_{V_j} are scalar parameters learned to estimate variance, for each user and each item, respectively. The α_p is a learned global precision parameter. Instead of modeling the likelihood of an exact rating value, the model defines a prediction interval within which the rating is expected to lie with a given confidence level p:

$$\left[\mu_{ui} - Z(p/2) \cdot \sqrt{\nu},\ \mu_{ui} + Z(p/2) \cdot \sqrt{\nu}\right]\,,$$

where $Z(p/2)$ denotes the quantile of the standard normal distribution corresponding to the two-tailed confidence level p. On the other hand, if the interval is fixed, its possible to obtain the confidence level by

$$P(r_{ui} = \mu_{ui} | \boldsymbol{\Theta}) = P(r_{ui} <= \mu_{ui} + \delta_r) - P(r_{ui} <= \mu_{ui} - \delta_r)\,,$$

for a small δ_r.

The model parameters, $\boldsymbol{\Theta}$, are learned by minimizing the negative log-likelihood, leading to the following loss function:

$$\mathcal{L}_{\text{CPMF}} = -\frac{1}{|\mathcal{R}|} \sum_{(u,i)\in\mathcal{R}} \log P(r_{ui} \mid \mathbf{U}, \mathbf{V}, \gamma_{\mathbf{U}}, \gamma_{\mathbf{V}}, \alpha_p)\,,$$

where $\mathcal{R}$ is the set of observed user-item interactions.

The second method variant is called CBPMF. In contrast to CPMF, CBPMF imposes prior distributions on the variance parameters. Specifically, γ_{Ui} and γ_{Vj} are assumed to follow Gamma distributions:

$$\gamma_{Ui} \sim \Gamma(a_g, b_g) \quad \text{and} \quad \gamma_{Vj} \sim \Gamma(a_g, b_g)\,,$$

where a_g and b_g are the shape and rate hyperparameters, respectively. Additionally, the latent factors themselves are constrained to follow Gaussian distributions:

$$\mathbf{U}_i \sim \mathcal{N}(\boldsymbol{\mu}_U, \boldsymbol{\Lambda}_U^{-1}) \quad \text{and} \quad \mathbf{V}_j \sim \mathcal{N}(\boldsymbol{\mu}_V, \boldsymbol{\Lambda}_V^{-1})\,.$$

To ensure the learned parameters respect these distributional constraints during training, CBPMF employs Gibbs sampling for parameter estimation. This probabilistic inference approach iteratively samples from the conditional posterior distributions of each parameter, allowing the model to enforce the Gamma and Gaussian priors. In contrast, using a gradient-based method such as SGD would not respect these constraints, and the parameters could drift outside their intended distributions.

4.4 LEARNED BETA DISTRIBUTIONS FOR CONFIDENCE IN COLLABORATIVE FILTERING

The proposed model, LBD (KNYAZEV; OOSTERHUIS, 2023), assumes that user–item interactions can be represented as a Beta distribution parameterized by the outputs of a matrix factorization-like architecture. Each user i and item j are represented by embedding vectors $\mathbf{U}_i, \mathbf{V}_j \in \mathbb{R}^d$. A scalar rating prediction is modeled by the mean $\mu_{ij} \in (0, 1)$ of a Beta distribution, while its confidence is encoded in a positive scalar $\nu_{ij} \in \mathbb{R}^+$.

The mean is computed using the cosine similarity between user and item embeddings:

$$\mu_{ij} = \frac{1}{2}\left(1 + \frac{\mathbf{U}_i^\top \mathbf{V}_j}{|\mathbf{U}_i||\mathbf{V}_j|}\right). \tag{4.16}$$

The confidence ν_{ij} is defined as the L2 norm of the element-wise sum of the user and item embeddings:

$$\nu_{ij} = |\mathbf{U}_i + \mathbf{V}_j|_2. \tag{4.17}$$

Using these two quantities, the shape parameters of the Beta distribution are computed by:

$$\alpha_{ij} = \mu_{ij} \cdot \nu_{ij}, \quad \beta_{ij} = (1 - \mu_{ij}) \cdot \nu_{ij}. \tag{4.18}$$

To improve flexibility, the model applies learned bias terms over users and items for both parameters:

$$\alpha'_{ij} = \max(a_0 + a_i + a_j + \alpha_{ij}, \varepsilon), \quad \beta'_{ij} = \max(b_0 + b_i + b_j + \beta_{ij}, \varepsilon), \tag{4.19}$$

where a_0, b_0 are global scalar biases, a_i, a_j, b_i, b_j are user– and item–specific learned biases, and ε is a small positive constant ensuring numerical stability.

The final predicted rating is the expected value of the Beta distribution rescaled to the original rating range:

$$\hat{r}_{ij} = \mu_{ij}(r_{\max} - r_{\min}) + r_{\min}. \tag{4.20}$$

To train the model, the continuous rating interval $[0, 1]$ is divided into (S) adaptive bins defined by learned user- and item-specific parameters. For each user (i) and item (j), the bin weights are computed as:

$$w_{ij}^{(s)} = \exp(\theta_i^{(s)} + \theta_j^{(s)}), \tag{4.21}$$

where $\theta_i^{(s)}$ and $\theta_j^{(s)}$ are the user and item adaptive bin embeddings for the (s)-th bin. These weights are normalized so that the cumulative sum defines the bin boundaries:

$$e_0 = 0, \quad e_{s+1} = \sum_{k=0}^{s} \frac{w_{ij}^{(k)}}{\sum_r w_{ij}^{(r)}}, \quad e_S = 1. \tag{4.22}$$

The probability mass for each bin is obtained from the Beta cumulative distribution function:

$$p_{ij}^{(s)} = \mathrm{CDF}_b(e_{s+1}; \alpha'_{ij}, \beta'_{ij}) - \mathrm{CDF}_b(e_s; \alpha'_{ij}, \beta'_{ij}). \tag{4.23}$$

The training loss is the negative log-likelihood of the bin corresponding to the ground-truth rating:

$$\mathcal{L} = -\frac{1}{N} \sum_{(i,j)} \log p_{ij}^{(s^*)} + \lambda |\mathbf{U}|_2^2 + \lambda |\mathbf{V}|_2^2, \tag{4.24}$$

where s^* is the index of the bin containing the normalized ground-truth rating, and the final term is L2 regularization over the embeddings.

All model parameters, including embeddings, biases, and adaptive bin weights, are learned using stochastic gradient descent. The Beta CDF is approximated numerically using a trapezoidal integration scheme. More details about optimization are presented in the next chapter, about machine learning techniques for collaborative filtering models.

4.5 SUMMARY

This chapter showed the concept of confidence and several approaches for estimating a model's confidence in individual predictions within recommender systems. We focus on methods that quantify the certainty of a model regarding a specific user–item outcome. The chapter examines three main families of techniques: OrdRec, which models ratings as ordinal probabilities; confidence-aware matrix factorization methods (CPMF and CBPMF), which incorporate uncertainty through variance modeling and Bayesian priors;

and learned beta distributions, which represent user–item interactions as Beta distributions parameterized by neural embeddings. Together, these methods highlight different probabilistic strategies for capturing prediction confidence and provide the foundation for evaluating and comparing confidence-aware recommendation models. The next chapter presents techniques for designing and optimizing machine learning models for collaborative filtering recommender systems.

Chapter 5

This section will review related works on confidence estimation incorporated into RecSys models. This review will elucidate how various studies have addressed the integration of confidence estimation within RecSys frameworks and highlight their contributions, which inform and shape the present study. Additionally, this section will identify current gaps in the literature regarding confidence estimation and its applications in RecSys models. Finally, a comparative table will be provided, summarizing the strengths and weaknesses of each reviewed approach by categorizing them based on their confidence estimation methods and rating prediction modeling techniques.

RELATED WORKS

Understanding and quantifying uncertainty is crucial in recommender systems for assessing the reliability of predictions and ensuring that recommendations strike a balance across multiple performance dimensions, including precision, coverage, novelty, and diversity. Mesas and Bellonín (2020) provided a taxonomy of techniques for measuring uncertainty in recommendation algorithms, particularly those based on matrix factorization (MF). They analyzed two strategies for estimating uncertainty and emphasized the importance of balancing recommendation quality across different metrics. Their approach to confidence estimation used an algorithm-independent strategy, evaluating methods implemented in RankSys [1]. They demonstrated that for nearest-neighbor algorithms, confidence estimation can be derived from the number of users who rated the same item, while for regression-based algorithms, it can be estimated using the standard deviation of predicted scores (MESAS; BELLOGÍN, 2020).

Their research provided key insights into independent confidence estimation. Subsequent studies in the following subsections 5.1 and 5.2 suggest that confidence should not be treated as an isolated measure but rather integrated into the model to influence recommendation behavior. Confidence estimation, when incorporated into ranking or filtering mechanisms, can impact precision, coverage, novelty, and diversity. For instance, high-confidence recommendations may improve precision but reduce novelty and diversity by favoring well-known items. Conversely, promoting lower-confidence recommendations could enhance novelty and diversity but might compromise precision.

[1] https://github.com/RankSys/RankSys

5.1 CONFIDENCE CALIBRATION AND GENERAL NEURAL NETWORK APPROACHES

Focusing on calibration metrics on RecSys models, Wonbin (2024) proposed two parametric calibration methods: Gaussian and Gamma calibration. The assumption is that the confidence follows a Gaussian or Gamma distribution. Then, given a dataset with explicit interactions, where values are either zeros or ones, the approach predicts whether the model is correct about the prediction. Zeros indicate that there was no interaction or dislike, while ones indicate an interaction and liking. They applied the proposed approach to Bayesian Personalized Ranking (BPR), Unbiased BPR, and Light Graph Convolutional Networks (KWEON, 2024a). Whilst this work proposes a calibration method for predicting whether a user would like an item, our focus is on rating prediction and ranking performance improvement.

Although not specifically in the context of RecSys, Moon et al. (2020) proposed an approach for NN model calibration. Then, they designed a loss function that captures the prediction confidence by reflecting the model's failure prediction based on the proportion of times a sample is correctly classified during training. This loss function is combined with cross-entropy loss to form the final objective (MOON et al., 2020).

Papadopoulos, Edwards, and Murray (2001) examined three methods for estimating the confidence of neural network models: Bayesian, maximum likelihood, and bootstrap approaches. Their objective was to predict confidence intervals for each model prediction. They assumed that the model's outputs follow a Gaussian distribution and identified two primary sources of uncertainty: the model's inherent limitations in learning patterns and errors in the data. Focusing on uncertainty estimation, their approach defines the probability of the target value falling within a given interval using the cumulative distribution function (CDF). Consequently, when the model makes accurate predictions, this probability should be higher, whereas incorrect predictions result in a lower probability (PAPADOPOULOS; EDWARDS; MURRAY, 2001). The approach can be categorized as a confidence-aware learning task and offers valuable insights for applications in RecSys, particularly when NN is utilized in this context. This focuses on modeling and evaluating confidence in probabilistic models, rather than improving RecSys performance. Our proposed approach focuses on improving the ranking performance of the RecSys model.

5.2 DISTRIBUTION-BASED CONFIDENCE INTEGRATION IN RECOMMENDER SYSTEMS

Confidence intervals are a powerful method for estimating the reliability of predictions in RecSys. Wang et al. (2018) propose incorporating confidence intervals into MF models to assess the uncertainty of prediction intervals. By modeling the distribution of ratings as following a Gaussian, they can compute the associated confidence levels for the desired prediction intervals. The authors evaluate their approach against commonly used recommender system baselines, employing RMSE to assess rating prediction accuracy (WANG et al., 2018a).

Koren and Sill (2011) present the OrdRec solution for confidence interval and point-

wise confidence estimation. The method is evaluated using the RMSE metric with Netflix, Y!Music-I, and Y!Music-II datasets. Wang et al. (WANG et al., 2018a) propose CPMF and CBPMF, incorporating confidence intervals into MF models to assess the uncertainty of prediction intervals. The solution employed RMSE to assess rating prediction accuracy over Movie Lens, Netflix, and Jester Joke datasets. Similarly, we employ RMSE as one of the metrics used, and MovieLens and Jester Joke are among the datasets for method evaluation. Despite presenting a good basis in confidence modelling for RecSys, the methods harm the models' accuracy (KOREN; SILL, 2011).

Himabindu, Padmanabhan, and Pujari (2018) proposed a transductive conformal and inductive conformal method combined with user-based and item-based collaborative filtering, referred to as the conformal MF method. In this method, user-based representation is defined as a row of the user-item ratings matrix, while item-based representation is defined as a column of this matrix. They used Nonnegative MF (NMF) due to its simplicity. For each rating possibility within the range [1, 5], a probability value is calculated, denoted as [p(1), ..., p(5)]. The highest probability value determines the predicted rating. Confidence in the prediction is defined as 1− the second-highest p-value (HIMABINDU; PADMANABHAN; PUJARI, 2018).

Gohari, Aliee, and Haghighi (2021) proposed a new Confidence-Based Recommendation (CBR) approach as a hybrid of entropy-based and trust-based CF systems, where the confidence level is leveraged to predict the recommendations. Their solution uses an associated confidence level to estimate the neighbors of the target user and item by calculating the entropy. Where there are User-level Local Confidence, User-level Global Confidence, Item-Level Local Confidence, and Item-Level Global Confidence, these confidence estimations are obtained by operations on the dataset, using Shannon entropy concepts (GOHARI; ALIEE; HAGHIGHI, 2018).

Knyazev and Oosterhuis (2023) presented an approach for RecSys that utilizes a Beta distribution to model the rating function, taking into account confidence. They utilized learned embeddings in a model to estimate ratings and confidence. The Beta distribution can be parameterized by mean and sample size. The mean was defined as the cosine similarity between the user and item embedding. The confidence was defined by three possibilities: the product of the L_2 norms of the embeddings (v_{ij}^{norm}), L_2 of the summation of embeddings (v_{ij}^{sum}), and the absolute of value of the product of the embeddings (v_{ij}^{dot}). They did experiments with the MovieLens 10M dataset, and the best result was found by choosing v_{ij}^{sum} as confidence estimation (KNYAZEV; OOSTERHUIS, 2023).

Coscrato and Bridge (2023) conducted an extensive empirical study comparing several classes of uncertainty estimators for recommender systems, including information-based, stability-based, error-based, distribution-based, and multinomial-based approaches, among them, OrdRec and CPMF. Their results revealed important concerns regarding the reliability and consistency of uncertainty estimates for rating prediction. Many models showed inconsistent correlations across different uncertainty–error evaluation metrics. For instance, a method could exhibit a strong Pearson correlation with prediction error while performing poorly under alternative measures such as Spearman correlation. Moreover, several estimators displayed weak overall correlations with most confidence evaluation

metrics. In addition to re-evaluating previously proposed uncertainty estimators, the authors also introduced two new approaches: ENSEMBLE and EB-Linear. ENSEMBLE estimates uncertainty by training N models with different random initializations, taking the mean of their predictions as the estimated rating and the standard deviation as the uncertainty level. EB-Linear, on the other hand, learns two models, one to predict ratings and another to predict prediction errors, where uncertainty is modeled as the sum of user- and item-specific bias terms. Although both methods were designed to be robust and computationally efficient, the experimental results indicated that they did not outperform simpler, information-based estimators in most evaluation metrics (COSCRATO; BRIDGE, 2023).

5.3 ALTERNATIVE APPROACHES LEVERAGING CONFIDENCE IN RECOMMENDER SYSTEMS

Some RecSys solutions that consider confidence have been proposed. For instance, Mazurowski (2013) shows the concept of confidence in RecSys and six different algorithms for confidence estimation. The article highlights the utility of confidence in improving RecSys performance by filtering recommendations and helping users choose them by displaying their confidence level. The algorithms shown are Support For User, Support For Item, Variability for Item, Resample, Resample Fast, and Inject Noise. Among these, the Resample-based algorithms provided the most useful estimates for confidence. They applied these confidence algorithms on a CF item-based MF with bias adjustment, which takes the user's tendency to give a rating based on the global average, item bias, and user bias. Despite its usefulness and representation of mindset, some of these algorithms present granularity issues, as they are unable to yield confidence for the pair (i,u), instead providing a constant for items or users. The Resample-based and injected noise still do not present advantages when applied to NN-based models because it raises computational resource issues (MAZUROWSKI, 2013). On the other hand, our work shows a lightweight granular approach.

Yang and Buettner (2021) proposed a new coregionalization kernel that captures the similarity between samples and the similarity between outputs in multi-output Gaussian Processes. Coregionalization is commonly used to capture relationships between several spatially distributed variables, while Gaussian Processes are a well-known model class that provides prediction uncertainty. They applied this kernel to the rating prediction problem with confidence estimation, evaluating the proposed solution regarding RMSE and MAE. The experiments used the MovieLens-1M, MovieLens-10M, and Jester Joke datasets (YANG; BUETTNER, 2021).

While it is not focused on confidence modeling but on its applicability, Shaikh et al. (2023) demonstrated an approach to augmenting data by leveraging rating predictions with high confidence as input data to retrain and refine the model. They performed experiments on the MovieLens 100K and MovieLens 1M datasets and evaluated a maximum margin MF (MMMF) model regarding RMSE and MAE (SHAIKH et al., 2024).

Neto, Costa, and Monzato (2018) show an approach (CoBaR) for grouping neighbors of the target user within a confidence level for CF RecSys with Hierarchical Clustering.

CoBaR leverages confidence estimation to identify the most suitable cluster containing the target item. For rating prediction, it calculates the final rating by averaging the mean rating of the confident cluster to which the target item belongs and the user's mean rating across all items. However, this approach requires the algorithm to run on the entire dataset to update the system for new user/item interactions. Additionally, the rating prediction function may not consider sufficient features, and parameterization can be challenging as more features are added (NETO; COSTA; MANZATO, 2018b). Whilst our proposed solution does not have these limitations.

As several previous works focus on confidence for the rating prediction task, ZHANG et al. (2016) focus on learn-to-rank methods. They propose a post-processing confidence estimation for collaborative filtering learn-to-rank recommenders (ZHANG; GUO; CHEN, 2016). However, this approach introduces an additional complexity, resulting in $O(m \cdot n^2)$. In comparison, a BPR-based approach has complexity of $O(m \cdot n)$, where m and n are the number of users and items, respectively. Wang et al. (2023) introduce the concept of epistemic uncertainty, which refers to the uncertainty in model parameters resulting from a lack of knowledge. The approach models Epistemic uncertainty by placing a distribution over the model weights and approximates it during inference using Monte Carlo Dropout, where multiple forward passes are performed with Dropout turned on. The model is also trained to produce a second output that quantifies the aleatoric uncertainty. As this approach is a neural network-specific architecture, it can not be generalized across different models, such as standard matrix factorization (WANG; KADIOĞLU, 2023). Additionally, deep learning model presents a considerable computational cost; meanwhile, their approach performs S forward passes for each prediction.

In the realm of optimization with denoised datasets, as presented in (ALVES; LEDENT; KLOFT, 2024), continuous confidence scores are derived from user and item metadata (e.g., review length or interaction patterns) to dynamically weight the loss function of the MF model. High-confidence ratings are prioritized during training, while low-confidence ones are down-weighted, reducing their influence on model predictions. The authors adapt the nuclear norm regularization by incorporating a weighted trace norm, which rescales rows and columns based on confidence scores. They experiment on synthetic and real-world datasets (e.g., MovieLens, Goodreads), assessing RMSE in rating prediction and Spearman correlation in ranking. Still, their optimization algorithm relies on truncated SVD computations during proximal steps, which scale as $O(r(m+n)^2)$ for rank-r approximations, raising scalability issues. Compared to ours, this work focuses on improving the RecSys model by leveraging confidence derived from metadata for denoising the dataset, whereas ours leverages confidence derived from the model to directly improve ranking performance.

5.4 SYNTHESIS AND RESEARCH GAPS

Given these prior related works, confidence enhances prediction accuracy, provides structural advantages in clustering, mitigates data sparsity, and balances diverse recommendation qualities, depending on the context and application. The variety of approaches reflects the complexity and importance of confidence estimation, indicating that future

RecSys solutions may continue to innovate by blending traditional and novel techniques to refine recommendation reliability and user experience. While the field of confidence estimation in RecSys has seen significant advancements, several areas need improvement. Addressing scalability, model complexity, and overconfidence in predictions is critical to developing more robust and adaptable models. Additionally, it is needed to ensure fine-grained confidence estimation and generalization across diverse datasets. Balancing the integration of traditional statistical methods with modern machine-learning techniques will likely play a significant role in overcoming current limitations and enhancing the effectiveness of RecSys. The Table 5.1 compares the presented works. The following gaps are identified across the related literature, with individual works exhibiting one or more of them:

- The absence of integrated confidence modeling in embedding-based recommender systems,
- The lack of confidence-aware GAT models,
- The lack of experimental understanding of existing distribution-based methods,
- The missing extension of confidence estimation to learning-to-rank,
- The accuracy–confidence trade-off that limits practical adoption.

Table 5.1: Comparison of various Solutions that can be used for RecSys.

Work/Author	**Confidence Approach**	**Modeling Technique**	**Limitations**
(WANG et al., 2018a)	Incorporates confidence intervals in matrix factorization models by modeling rating distribution as Gaussian.	MF with confidence intervals to assess prediction uncertainty.	Struggles with capturing complex user-item interactions due to MF limitations.
(NETO; COSTA; MANZATO, 2018b)	Groups neighbors of the target user within a confidence level using Hierarchical Clustering.	Hierarchical Clustering for CF RecSys	Requires running on the entire dataset for updates, which challenges evaluation across different dataset splits and introduces difficulty in parameterization with added features.
(HIMABINDU; PADMANABHAN; PUJARI, 2018)	Defines confidence as 1 minus the second-highest p-value from predicted ratings.	CMF using Transductive and Inductive Conformal methods.	Limited by the inability of MF to capture complex non-linear relationships.
(MESAS; BELLOGÍN, 2020)	Algorithm-independent confidence estimation using nearest-neighbor and regression algorithms	MF based algorithms and nearest-neighbour.	Lack of influence of confidence on model behavior and outcomes.
(MOON et al., 2020)	Specialized loss functions encourage reliable prediction confidence in DL models.	(Not a RecSys solution) DL models with confidence-aware loss functions.	High confidence in incorrect predictions, requiring careful loss function design.

Work/Author	Confidence Approach	Modeling Technique	Limitations
(YANG; BUETTNER, 2021)	Uses a coregionalization kernel to estimate confidence within Gaussian Processes.	Multi-output Gaussian Processes with a coregionalization kernel.	Requires computational resources for Gaussian Processes and may face scalability issues with large datasets.
(GOHARI; ALIEE; HAGHIGHI, 2018)	Confidence level estimated using Shannon entropy-based approach.	Hybrid entropy-based and trust-based CF systems.	Results do not meet necessary standards.
(KNYAZEV; OOSTERHUIS, 2023)	Confidence estimated using Beta Distribution with learned embeddings.	Assumes a rating following a Beta Distribution	Assumes a suboptimal beta distribution for user ratings.
(COSCRATO; BRIDGE, 2023)	Systematic evaluation and proposal based on ensemble and error prediction.	Ensemble and error prediction with MF	Additional complexity for running N models; the uncertainty/confidence from the models has a low relationship with the metrics.
(KOREN; SILL, 2011)	Confidence estimation.	Normal distribution constraint over the ratings.	The method harms the model accuracy.
(SHAIKH et al., 2024)	Confidence for data augmentation in Maximum Margin Matrix Factorization (MMMF).	MMMF approach with high-confidence rating predictions as input.	Limited evaluation on larger datasets.
(PENG; SUGIYAMA; MINE, 2024)	N/A	GNNs-based approaches for modeling complex relationships.	Graph construction dependency affects performance and scalability.

Work/Author	**Confidence Approach**	**Modeling Technique**	**Limitations**
(HUANG et al., 2024)	N/A	Broad collaborative filtering with neighbor encoding using cosine similarity.	Generalization is affected by the neighbor definition and the ambiguity in dataset splitting.
(KWEON, 2024a)	Assumes a Gaussian or Gamma distribution for the confidence.	Bayesian Personalized Ranking, Unbiased BPR, and Light Graph Convolutional Networks.	Parametric model assumption.
(MOON et al., 2020)	Loss function for NN model calibration	NN model	Not comparable with RecSys approaches
(ZHANG; GUO; CHEN, 2016)	Post processing method	Matrix factorization	Additional complexity
(WANG; KADIOĞLU, 2023)	Post processing method	Neural network	Additional complexity
Ours	Distribution-based	Any embedding-based model	Discussed in sections 7.7 and 8.6

5.5 SUMMARY

This section reviews related works on confidence estimation incorporated into RecSys models. It elucidated how various studies addressed the integration of confidence estimation within RecSys frameworks and highlighted their contributions, which informed and shaped the present study. Additionally, this section identified current gaps in the literature regarding confidence estimation and its applications in RecSys models. Finally, a comparative table was provided, summarizing the strengths and weaknesses of each reviewed approach by categorizing them based on their confidence estimation methods and rating prediction modeling techniques. Given the current limitations of the highlighted literature, the next chapter presents a proposal for confidence integration in the rating prediction and pairwise learn-to-rank tasks.

Chapter 6

This chapter shows the proposal variations and formal definitions. This chapter details the approach applied for rating prediction and learn-to-rank methods. This chapter is then divided into two parts, each representing one proposal. The first is focused on rating prediction, and the second on pair-wise learn-to-rank.

CONFIDENCE FOR COLLABORATIVE FILTERING MODELS

Note that this dissertation introduces two proposals for integrating confidence into CF models: the first focuses on rating prediction, and the second on pairwise ranking tasks. Accordingly, this chapter is divided into three sections. Section 6.1 presents the proposed model based on the GAT architecture, referred to as Deep Graph Attention Networks (DGAT). This model serves as a core component of the complete confidence integration method. Section 6.2 then introduces the full approach for incorporating confidence into embedding-based models for rating prediction, where DGAT is employed. Finally, Section 6.3 describes the method for integrating confidence into collaborative filtering models for pairwise learn-to-rank, completing the second proposal

6.1 DEEP GRAPH ATTENTION NETWORKS MODEL

We aim to explore confidence integration with GAT-based methods. First, we evaluate the performance of a simple implementation without confidence integration. The DGAT utilizes the GAT implementation from VELIčKOVIć et *al.* (2018) to learn a second-level latent space representation of users and items. The user and item identification (ID)s feed this model, which returns the prediction for the pair $(u, i) \mapsto \hat{r}$. This is achieved by passing a partial graph, consisting of a subset of user-item pairs, to the model, where a batch of edges is selected from the original graph. The GAT(features, edge) implementation receives a node feature matrix and the partial graph (edge - E) (VELIčKOVIć et al., 2018), where this partial graph is a batch of tuples. We pass the partial graph with only the edges, without weights (ratings). We use the embedding matrix $\mathbf{E}$ as the feature matrix, which is the concatenation of users' and items' embeddings. Thus, we define $\mathbf{E}$

as the initial latent representation for users and items, and $\mathbf{e}'$ as the result after applying the GAT layer,

$$\mathbf{e}' = \text{GAT}(\mathbf{E}, E)\,. \tag{6.1}$$

After this transformation, $\mathbf{e}'$ is then passed to a feed-forward neural network with two layers, as expressed as follows:

$$\begin{aligned}\mathbf{e}'' &= \text{Dropout}(\text{LeakyReLU}\left(\mathbf{W}_1 \begin{bmatrix}\mathbf{e}'_\mathbf{u} \\ \mathbf{e}_\mathbf{i}\end{bmatrix} + \mathbf{b}_1\right)) \\ \hat{\mathbf{r}} &= \mathbf{W}_2\mathbf{e}'' + \mathbf{b}_2\,,\end{aligned}$$

where W's and b's are the weights and bias matrices of the feedforward NN, $\mathbf{W_1} \in R^{2d_{emb} \times d_{emb}}$, $\mathbf{W_2} \in R^{d_{emb} \times 1}$. This model is optimized using the same loss function of MF. However, this model has no confidence incorporated in its design.

6.2 PROBABILISTIC RECOMMENDER GRAPH ATTENTION NETWORKS

We combine the proposed confidence modeling from CPMF with the rating prediction from DGAT to produce PRGAT. Thus, the rating is estimated using the DGAT implementation. We optimize the probability of observing a rating, given a pair (u, i), as the same in CPMF, by assuming the ratings follow a normal distribution. Then, we set up a Normal distribution with mean μ and variance σ^2_{ui}, where μ is inferred as the predicted rating, $\mu_{ui} = \hat{r}$, using the DGAT model. We define the standard deviation of the distribution as

$$\sigma_{\mathbf{ui}} = \frac{1}{\sqrt{\gamma_\mathbf{u} \cdot \gamma_\mathbf{i} \cdot \nu'}}\,, \tag{6.2}$$

where γ_u, γ_i, and ν' are learned parameters for users, items, and global, respectively. Therefore, the model's confidence is defined as the probability of observing the predicted rating,

$$P(r_{ui} = \hat{r}|\Theta) = P(r_{ui} \leq \hat{r} + \delta_r) - P(r_{ui} \leq \hat{r} - \delta_r)\,, \tag{6.3}$$

where $P(r_{ui} \leq q) = \Phi(q) = CDF(q)$, using the cumulative distribution function of the Normal distribution.

However, the final goal is to maximize the probability of observing the current observed rating and reduce the MSE error as well, changing the loss function to

$$\mathcal{L}_{PRGAT} = \frac{1}{|\mathcal{R}|} \sum_{(u,i)\in\mathcal{R}} \left(-\log(P(r_{ui} = \mu_{ui}|\Theta)) + \lambda \cdot \text{MSE}(r_{ui}, \mu_{ui})\right)\,, \tag{6.4}$$

where λ is a tunable parameter. Figure 6.1 illustrates the overall model architecture, comprising each module, its operation details, and the data flow.

Unifying graph-based relational learning with probabilistic uncertainty modeling offers some theoretical advantages over traditional confidence-aware and embedding-based recommender approaches. Unlike prior methods that treat confidence estimation as a post-hoc adjustment to deterministic predictions, PRGAT integrates uncertainty directly

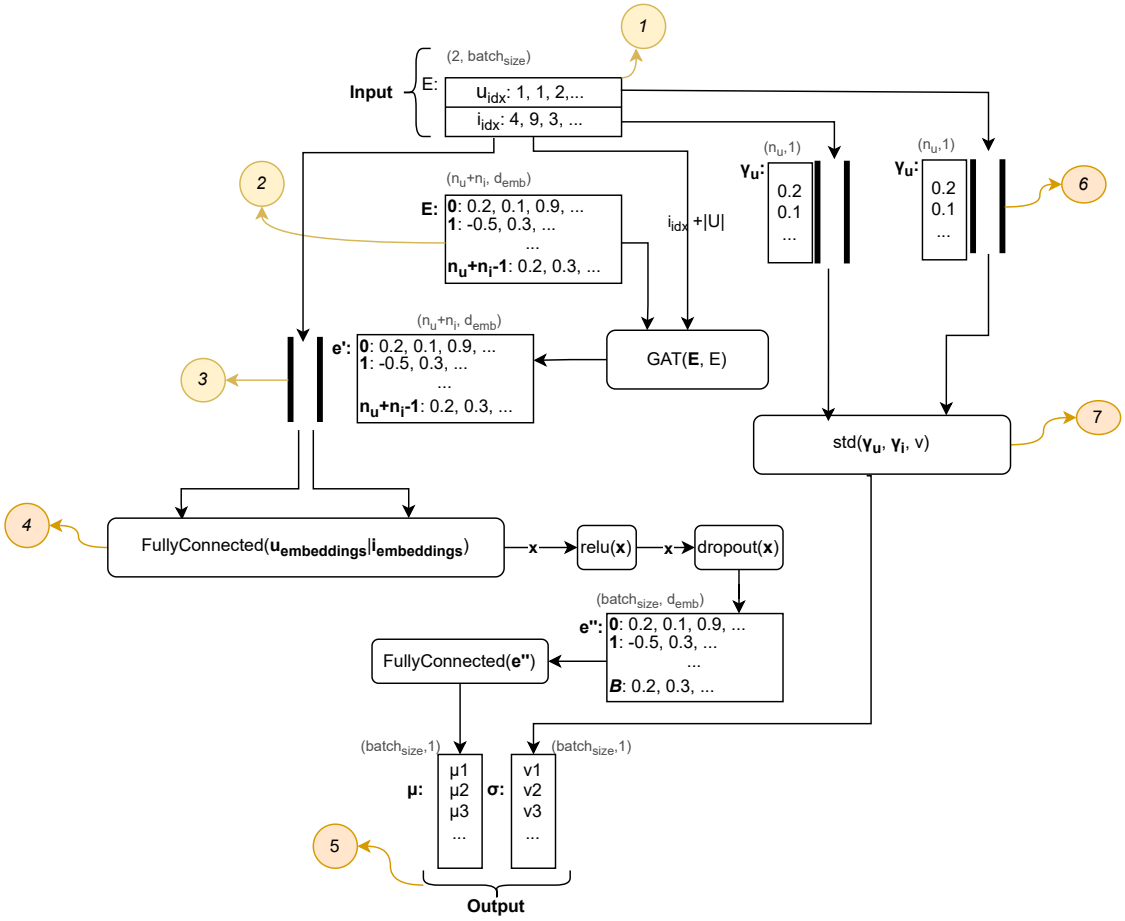

Figure 6.1: Proposed model integrated with confidence, PRGAT, architecture. The arrows define the order of transformations and data transfer. (1) represents the data input, which is the subgraph, that is, a batch of edges, $E \in R^{(2,batch_{size})}$. (2) represents the node embeddings for both users and items, $\mathbf{E} \in R^{(n_u+n_i,d_{emb})}$. (3) represents the operation of node embedding gathering using advanced indexing. For each node ID, the operation returns its corresponding embedding from the shared embedding matrix, $\mathbf{e}'$. (4) applies a fully connected neural network over the concatenation of $u_{embedding}$ and $i_{embedding}$, with d_{emb} neurons. (5) is the model output, μ and σ, which represent mean and standard deviation to parameterize a normal distribution function. (6) advanced index retrieval over γ_u using users IDs. (7) applies Equation 6.2 over the retrieved $\gamma_\mathbf{i}$, $\gamma_\mathbf{u}$, and ν. $\boldsymbol{B}$ is the batch size.

into the representation learning process. By parameterizing a probabilistic distribution over ratings using graph-based embeddings, the model captures both the expected preference (μ_{ui}) and its reliability (σ_{ui}) within a single optimization framework. This joint modeling enables the network to learn confidence-aware latent spaces that inherently encode the uncertainty of user–item interactions. Through the attention mechanism of GAT, the proposed model explicitly propagates user–item dependencies along the interaction graph, allowing node representations to reflect higher-order neighborhood influence. This theoretically improves robustness to sparse data by leveraging indirect relationships (e.g., users connected through similar items) and mitigates cold-start effects.

By assuming that ratings follow a normal distribution and learning the variance term as a function of user and item-specific precision parameters (γ_u), (γ_i), and a global scaling factor (ν), the model introduces a natural probabilistic regularization. This prevents overconfident predictions and encourages well-calibrated estimates of uncertainty. The negative log-likelihood term in the loss function ensures statistical consistency between predicted means and observed ratings.

The proposed loss function ($\mathcal{L}_{PRGAT}$) combines the likelihood-based objective with the mean squared error term, providing a principled way to balance predictive accuracy and uncertainty calibration. The tunable coefficient λ controls the trade-off between minimizing reconstruction error and maximizing probabilistic fidelity, allowing for customization to achieve better generalization across datasets. Moreover, the combination of the MSE and log-likelihood terms supplies a stronger and more stable gradient signal for backpropagation, as the MSE component reinforces convergence toward accurate predictions while the likelihood term refines the probabilistic consistency of the confidence estimates.

The probabilistic design of PRGAT is architecture-agnostic, as the same confidence integration scheme can be extended to other graph-based or embedding-based models. This theoretical modularity facilitates adaptation to alternative graph neural architectures or other probabilistic distributions without altering the training dynamics. The source code for this work is publicly available on GitHub at <https://github.com/joel021/distribution-conf-rating-recsys.git>. However, as this solution is focused on the rating prediction task, a broader application in learn-to-rank remains open. Therefore, we also propose a solution for the pair-wise method, which is generalizable for the list-wise method, in the next section.

6.3 GAUSSIAN UNCERTAINTY-AWARE LEARNING-TO-RANK

Similar to prior works, such as OrdRec, CPMF (KOREN; SILL, 2011; WANG et al., 2018a), and PRGAT, for each pair, (u, i), we model the item's relevance as a Gaussian:

$$s_{ui} \sim \mathcal{N}\left(\mu_{ui},\ \sigma_{ui}^2\right) , \tag{6.5}$$

where the mean score (μ_{ui}) is given by any embedding-based model, like MF-based, graph-based, or neural networks:

$$\mu_{ui} = f(\tilde{e}_u, \tilde{e}_i) . \tag{6.6}$$

The variance of the score distribution is defined via precision:

$$\pi_{ui} = e^{\alpha}\,\gamma_u\,\gamma_i\,, \tag{6.7}$$

where $\gamma_u > 0$ is a learnable user-specific variance parameter, $\gamma_i > 0$ is a learnable item-specific variance parameter, $\alpha > 0$ is a global scaling parameter, and e is euler constant. Thus:

$$\sigma_{ui}^2 = \frac{1}{\pi_{ui}} = \frac{1}{e^{\alpha}\,\gamma_u\,\gamma_i} \tag{6.8}$$

The optimization of this model is via a pairwise ranking approach using a loss based on BPR loss. For each user u, with a positive item i^+ and a negative item i^-, define:

$$\Delta_{u,i^+,i^-} = s_{ui^+} - s_{ui^-}\,. \tag{6.9}$$

Since both s_{ui^+}, s_{ui^-} are Gaussian and independent,

$$\Delta_{u,i^+,i^-} \sim \mathcal{N}\left(\mu_{ui^+} - \mu_{ui^-},\ \sigma_{ui^+}^2 + \sigma_{ui^-}^2\right)\,. \tag{6.10}$$

The probability of correct ranking is

$$P(s_{ui^+} > s_{ui^-}) = \Phi\left(\frac{\mu_{ui^+} - \mu_{ui^-}}{\sqrt{\sigma_{ui^+}^2 + \sigma_{ui^-}^2}}\right)\,, \tag{6.11}$$

where Φ is the cumulative distribution function (CDF) of the standard normal distribution. Therefore, the training loss is

$$\mathcal{L} = -\frac{1}{|\mathcal{D}|}\sum_{(u,i^+,i^-)\in\mathcal{D}} \log(P(s_{ui^+} > s_{ui^-}))\,. \tag{6.12}$$

This is a Gaussian BPR loss, analogous to the logistic BPR but with a normal noise assumption.

At inference, the model outputs the mean score μ_{ui}, point estimate of preference strength, and a confidence measure. This way, we define the confidence in the predicted score as

$$c_{ui} = \Phi\left(\frac{\mu_{ui}}{\sigma_{ui}}\right)\,. \tag{6.13}$$

This aligns with the defined loss function that pushes low item scores to be below zero and high item scores to be above zero.

Previous distribution-based models typically leave the variance term unconstrained, allowing it to freely increase or decrease as long as the mean of the distribution approximates the true score. In contrast, our formulation explicitly ties the variance (or standard deviation) to the optimization objective. Specifically, the standard deviations of both positive and negative item scores are incorporated as normalization terms within the probability expression used in the loss function. This design aims to directly influence the learning dynamics through variance.

Another interesting aspect is that our approach is generalizable for other learn-to-rank methods, such as RankNet (BURGES et al., 2005), where its loss function is:

$$\mathcal{L}_{\text{RankNet}} = -\sum_{(i,j)} [\bar{P}_{ij} \log P_{ij} + (1 - \bar{P}_{ij}) \log(1 - P_{ij})], \tag{6.14}$$

where $P_{ij} = P(s_{ui^+} > s_{ui^-})$ and $\bar{P}_{ij}$ is the ground-truth preference (1 if i is preferred over j, otherwise 0). However, we use the loss function as defined in Equation 6.12 for our experiments. The implementation code of this approach is publicly available at <https://github.com/joel021/dist-conf-learn-rank-recsys.git>. The implementation is written in Python and primarily utilizes the libraries PyTorch, pandas, and NumPy.

6.4 SUMMARY

This chapter investigates confidence estimation in embedding-based recommender systems and evaluates both prior distribution-based approaches and graph-based architectures. The chapter introduces PRGAT, which combines DGAT's graph-based representation learning with CPMF-inspired probabilistic modeling to jointly predict ratings and estimate uncertainty without significantly harming accuracy. The chapter also extends confidence estimation to learning-to-rank tasks by modeling item relevance as a Gaussian distribution and optimizing a Gaussian BPR objective that accounts for both mean scores and variance. This approach generalizes to any embedding-based model, yielding calibrated confidence estimates for ranking. Together, the proposed methods demonstrate how probabilistic modeling can be integrated directly into representation learning. The next chapter presents the experimental evaluation of the proposal, focusing on rating prediction.

Chapter 7

This experimental evaluation evaluates the "Exploiting Distribution-Based Confidence Integration in Graph Neural Network" variation of the proposed solution. This chapter will evaluate the prior distribution methods for confidence integration, as well as the proposed approach that integrates confidence into GAT-based models focused on rating prediction, regarding regression, ranking metrics, and confidence-error correlation.

EXPERIMENTAL EVALUATION 1: ASSESSING THE RATING PREDICTION METHOD

Our primary goal is to examine existing confidence estimation methods and propose an approach to estimating confidence for embedding-based recommender systems models without compromising performance. This evaluation is divided into two parts: the experimental evaluation of prior distribution-based methods for confidence estimation in recommender systems, and the evaluation of the proposed method, PRGAT. This chapter begins with the methodology (Section 7.1), which outlines the procedures used to evaluate both the prior confidence models and the proposed approach. It then presents the models, validation strategy, datasets, and metrics. The following sections provide a detailed description of these components, along with the results, discussion, and limitations.

7.1 METHODOLOGY

The experiments occur in three parts. The first part comprises the distribution-based model comparisons, OrdRec(KOREN; SILL, 2011), CPMF (WANG et al., 2018a), CBPMF (WANG et al., 2018a), and LBD (KNYAZEV; OOSTERHUIS, 2023). After analyzing their results in terms of accuracy, rank quality, and the relationship with absolute error, we perform new experiments with models that exclude confidence, MF, and DGAT, thereby closing the second part. The third part experiments with the proposed approach against its baselines and against the previous distribution-based approaches. These approaches are described in the Chapter 2, sections 2.4, 2.5, and Chapter 4.

For the ranking evaluation, we used the NDCG and MAP. We set the candidate items for each user as the observed items from the subset, plus 72 negative items. This number

is set according to the available computational resources and time. Since this number was kept consistent across all methods, it does not affect the fairness of the comparison or the relative performance evaluation among models. Note that the observed set of items contains both positive and negative items. The additional negative items for a specific user are defined as the set of items that have been interacted with by at least one user but not rated by the target user. This constructed subset of candidates is ordered by the ranking scores or ratings produced by the models. We evaluate the models in the evaluation and test subsets; however, we bring only the performance results from the test subset.

For the experimental setup, the embedding sizes of the models were set to 512, 20, 20, and 522 for OrdRec (OrdRec), CPMF, CBPMF, and LBD, respectively. The embedding sizes are taken from the original papers that introduced these methods. The embedding size is set to 64, and embeddings are initialized using the Xavier Uniform method for PRGAT, MF, and DGAT. This embedding size is the result of empirical observations on the performance of these models, where no improvement was observed when the size was increased beyond 64. Additionally, our empirical analysis shows that the embeddings become highly sparse when using large embedding sizes, such as 512, for these models.

All the aforementioned models were trained using the Adam optimization algorithm with a learning rate of 0.001, except for CBPMF. We employed Gibbs sampling for optimizing CBPMF, as described in (WANG et al., 2018a). The methods were implemented in PyTorch using the Python programming language. Early stopping was applied as a regularization strategy across all training procedures, with a patience threshold of 5 epochs without improvement in the evaluation error. A patience of 5 is set based on empirical observations showing that the models stopped improving after 2 epochs without progress; therefore, an additional 3 epochs were added as a margin.

We evaluate model performance using a 5-fold Monte Carlo cross-validation procedure. In the first fold, we apply a sorted split, while the remaining four folds use random splits. This choice avoids unnecessary repetition: performing Monte Carlo cross-validation with multiple sorted splits is generally redundant because sorting produces very similar distributions across folds. This cross-validation procedure is applied to all models and datasets. The datasets used are described in the next section.

7.2 DATASETS

The experiments utilize three publicly available datasets: Amazon Movies and TV, Jester Joke (GOLDBERG et al., 2001), and MovieLens 1M (HARPER; KONSTAN, 2015). These datasets were selected based on the related works to cover distinct recommendation scenarios. They are all tabular, containing user ID, item ID, the explicit feedback rating, and timestamp columns.

The Amazon Movies and TVs dataset [1] represents a large-scale movies and television domain. It spans user–item interactions covering models of user feedback on purchase and rating behavior. Each interaction includes a discrete rating (typically 1–5 stars) and the corresponding user–item context (user ID, product ID, timestamp). In this version, the

[1] Available at: <https://cseweb.ucsd.edu/~jmcauley/datasets/amazon/links.html>

Table 7.1: Summary of the datasets after preprocessing and filtering steps. n_u, n_i, and n_{inter} are the number of users, items, and interactions, respectively. $train_{ratio}$ denotes the proportion of the dataset used for training. The same definition applies to $eval_{ratio}$ and $test_{ratio}$.

Dataset	n_u	n_i	n_{inter}	sparsity	$train_{ratio}$	$eval_{ratio}$	$test_{ratio}$
Amazon Movies TVs	1316	7164	1791602	0.9998	0.7243	0.1378	0.1378
Jester Joke	2498	99	1800475	0.2720	0.7999	0.1000	0.1000
Movie Lens 1M	6040	3416	999611	0.9515	0.7478	0.1260	0.1260

original number of users and items is $2,088,620$ and $200,941$, respectively. The number of interactions is $4,607,047$. This results in an extremely large and sparse dataset, with a sparsity level of 0.9999.

The Jester Joke dataset [2] represents a large-scale real-world humor recommendation domain with explicit user feedback in the form of continuous ratings. It contains approximately 4.1 million user–item interactions, where each interaction corresponds to a real-valued rating ranging from –10.00 to +10.00, assigned by 7,699 users to 158 jokes. The dataset was collected between April 1999 and May 2003, reflecting diverse user preferences and perceptions of humor across time. The data is organized as a 7,699 × 159 matrix, where each row corresponds to a user and each column to a joke, with the first column indicating the number of jokes rated by that user. A value of 99 denotes a missing or unrated joke. Unlike traditional discrete rating datasets, Jester's continuous-valued feedback enables the study of fine-grained preference modeling and nonlinear rating behaviors. The dataset exhibits moderate sparsity relative to its dense rating structure, making it a valuable benchmark for collaborative filtering and recommender system research.

The MovieLens 1M dataset[3] is a widely used benchmark in recommender systems research, especially for evaluating collaborative filtering algorithms in the context of movie preferences. It contains 1,000,209 explicit ratings ranging from 1 to 5, provided by 6,040 users over 3,416 movies. Due to the large number of possible user-item pairs, the dataset exhibits a high sparsity level of 0.9553. After the preprocessing and splitting process, the resultant dataset's sample quantity has decreased, and its sparsity has changed. The Table 7.1 describes the datasets after this process.

After choosing the datasets, we perform data preparation processes to prepare the data for the experiments. The next section describes these processes.

7.3 DATA PREPARATION

Before splitting, we apply a multi-step data preparation process to ensure consistent and valid evaluation. We first filter out users and items with fewer than five interactions to reduce sparsity. Then, for each user, we perform a user-wise split: interactions are

[2]Available at: <https://goldberg.berkeley.edu/jester-data/>

[3]Available at <https://grouplens.org/datasets/MovieLens/1m/>

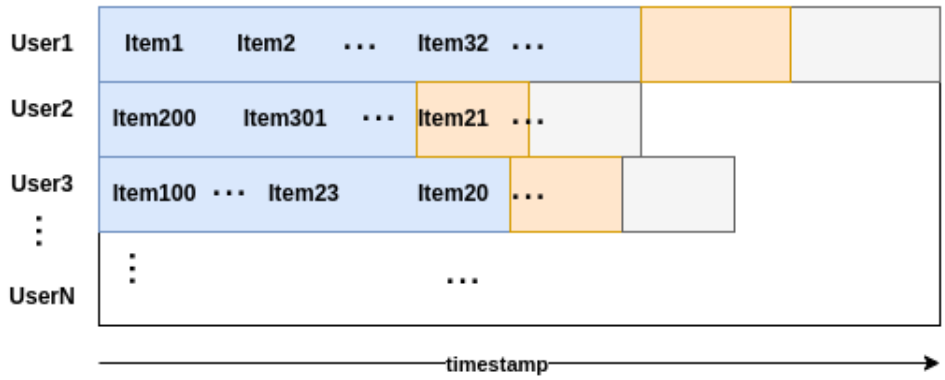

Figure 7.1: Sorted data splitting strategy. The timestamp is used to sort the interactions with the items when using the sorted splitting strategy. The blue part represents the train split, the orange represents the validation split, and the gray part represents the test split.

split by temporal order when timestamps are available (for sorted splits) or shuffled beforehand (for random splits). After the initial division, we further adjust the partitions to guarantee that all users and items appearing in the validation or test sets also have adequate representation in the training set. Items that appear only in the test set or have insufficient training interactions are partially moved to the training portion, and users with too few training interactions are removed entirely from both sets.

All datasets are ultimately divided into three subsets—training, validation, and testing—with target ratios of approximately 75%, 12.5%, and 12.5%, respectively, as illustrated in Figure 7.1. In sorted splits, each user's interactions are ordered by timestamp before partitioning. In random splits, interactions are shuffled. The sorted split introduces a higher level of difficulty for the model by better mimicking real-world temporal dynamics; however, because its folds would be nearly identical across repeated runs, we apply it only to the first fold and use random splits for the remaining folds.

After each execution of one fold in the cross-validation, obtain the performance metrics of the model over the dataset split, for each split. The next subsection presents the metrics used.

7.4 EVALUATION METRICS

The evaluation metrics include:

- Prediction Accuracy Metrics: RMSE.
- Ranking Metrics: NDCG@10, NDCG@3, MAP@10, and MAP@3.
- Confidence-error correlation: Pearson correlation.

After all folds, executions across all models and datasets, we evaluate the results.

Table 7.2: Metrics of the proposed and baseline models in the test split of Amazon Movies TVs.

Models	RMSE	MNDCG@10	MAP@10	MNDCG@3	MAP@3
ORDREC	1.634±0.626	0.176±0.002	0.159±0.003	0.165±0.003	0.157±0.002
CPMF	4.244±0.313	0.095±0.002	0.093±0.002	0.075±0.002	0.073±0.004
CBPMF	1.538±0.019	0.175±0.004	0.159±0.003	0.165±0.006	0.157±0.006
LBD	1.499±0.015	0.172±0.001	0.154±0.002	0.164±0.002	0.156±0.002
PRLIGHTGCN	1.057±0.02	**0.592**±0.011	**0.541**±0.011	**0.62**±0.014	**0.618**±0.015
PRGAT	0.948±0.007	0.154±0.036	0.155±0.028	0.133±0.053	0.136±0.05
MF	0.967±0.011	0.299±0.015	0.265±0.014	0.345±0.018	0.337±0.018
DGAT	0.95±0.008	0.144±0.021	0.149±0.016	0.112±0.03	0.117±0.03
LIGHTGCN	1.063±0.022	0.573±0.011	0.526±0.011	0.598±0.017	0.597±0.018

7.5 RESULTS

The results analysis begins by comparing the prior distribution-based confidence estimation methods in RecSys. To support this comparison, we present tables with the evaluation metrics, one per dataset, containing all prior methods together. To further validate our proposed method, we compare it against prior approaches and additional baselines. Therefore, we include a second table for each dataset, which presents the same evaluation metrics for both the proposal and baseline models. Splitting the results into two tables enables these two levels of comparison. Thus, for each dataset, we provide two tables for analysis in sequence: one for the prior distribution-based methods and another for the proposed method, along with the baselines. Although the prior distribution-based methods can also be viewed as baselines, here we consider as baselines only the models without confidence integration, which are expected to yield the highest performance metrics due to their design focus. We also evaluate the confidence output of the methods, including their distribution and the correlation between confidence and error.

Table 7.2 reports that LBD achieved the lowest RMSE in the Amazon Movies TVs dataset, indicating the most accurate rating predictions. In contrast, OrdRec and CBPMF obtained the best results in ranking-based metrics. Overall, the results indicate that no single distribution-based model consistently outperforms the others across all metrics. LBD improves rating accuracy, OrdRec, and CBPMF demonstrate better ranking capability. Yet in the Table 7.2, PRGAT achieved the lowest RMSE, slightly outperforming DGAT and MF in rating prediction. However, MF clearly surpassed all other models in ranking-based metrics. This contrast indicates that while PRGAT provides a more accurate estimation of ratings, it struggles to effectively prioritize top-ranked items compared to the MF baseline. These results suggest that, in this dataset, explicit relational information from the graph structure contributes less to ranking quality, and traditional MF remains the most effective approach for recommendation ordering. Still, PRGAT outperforms all letter distribution-based methods in all evaluated metrics for this dataset.

For the Jester Joke dataset, Table 7.3 shows that CPMF outperforms the other methods, followed by OrdRec, CBPMF, and LBD. CPMF excels in ranking quality, and

Table 7.3: Metrics of the proposed and baseline models' performances in the test split of the Jester Joke dataset.

Models	RMSE	MNDCG@10	MAP@10	MNDCG@3	MAP@3
ORDREC	12.624±0.137	0.57±0.064	0.292±0.023	0.479±0.069	0.279±0.052
CPMF	9.993±11.269	0.616±0.166	0.286±0.056	0.56±0.214	0.358±0.144
CBPMF	6.472±0.355	0.559±0.094	0.256±0.038	0.471±0.087	0.268±0.05
LBD	11.45±0.026	0.541±0.108	0.222±0.059	0.454±0.091	0.225±0.045
PRLIGHTGCN	4.87±0.189	**0.772**±0.028	**0.447**±0.015	**0.752**±0.019	**0.567**±0.018
PRGAT	4.796±0.141	0.708±0.08	0.418±0.047	0.692±0.091	0.522±0.073
MF	4.561±0.049	0.639±0.153	0.318±0.065	0.605±0.17	0.407±0.111
DGAT	4.859±0.024	0.678±0.121	0.29±0.054	0.676±0.11	0.397±0.085
LIGHTGCN	5.125±0.127	0.708±0.023	0.381±0.006	0.678±0.009	0.482±0.008

Table 7.4: Proposed and baseline models' performance over the test split of the Movie Lens 1M dataset.

Models	RMSE	MNDCG@10	MAP@10	MNDCG@3	MAP@3
ORDREC	1.538±0.399	0.201±0.007	0.164±0.009	0.187±0.008	0.162±0.01
CPMF	1.344±0.152	0.088±0.006	0.096±0.009	0.055±0.015	0.057±0.016
CBMF	1.364±0.017	0.188±0.045	0.168±0.037	0.177±0.043	0.169±0.038
LBD	1.247±0.014	0.211±0.002	0.172±0.003	0.199±0.002	0.174±0.005
PRLIGHTGCN	0.857±0.009	0.611±0.052	0.565±0.047	0.654±0.067	0.661±0.068
PRGAT	0.872±0.008	0.48±0.032	0.452±0.025	0.499±0.046	0.517±0.042
MF	0.882±0.009	0.457±0.041	0.435±0.036	0.495±0.056	0.505±0.055
DGAT	0.881±0.011	0.422±0.029	0.409±0.03	0.386±0.033	0.419±0.036
LIGHTGCN	0.848±0.009	**0.612**±0.05	**0.567**±0.048	**0.655**±0.064	**0.663**±0.065

CBPMF achieves the best prediction accuracy. This suggests that CBPMF models the underlying rating distribution more precisely, even though it does not necessarily produce the best ranking of items. Still in the Jester Joke dataset, Table 7.3 shows that the PRGAT model clearly outperforms all baselines across the ranking-oriented metrics. In terms of rating prediction accuracy, MF achieved the lowest RMSE, followed by PRGAT. Overall, these results indicate that PRGAT effectively leverages graph-based relational information and probabilistic modeling to enhance the quality of recommendation rankings while maintaining competitive performance in rating prediction. Finally, we compare the baseline and proposed models with the prior distribution-based models. The results show that PRGAT consistently outperforms all distribution-based models across all evaluation metrics. Moreover, even the baseline methods surpass the distribution-based approaches, indicating that the latter's probabilistic formulations may introduce optimization challenges or modeling constraints that hinder overall performance.

Among the evaluated methods in the Table 7.4, LBD achieved the best overall results, outperforming the other models across all evaluated metrics. CBPMF and OrdRec followed with comparable ranking performance, while CPMF showed the weakest results

across all metrics. Analysing Table 7.4, the PRGAT model consistently achieved the best results across all evaluation metrics. Although the performance is very close in terms of RMSE, the consistent improvements across all metrics suggest that incorporating probabilistic relational information through graph attention mechanisms enables PRGAT to better capture user–item dependencies and model nuanced preference patterns in the MovieLens 1 M dataset. It is worth noting that PRGAT also outperforms all distribution-based models across every evaluated metric on the MovieLens 1M dataset.

The confidence estimation scores are also evaluated. The figures 7.2, 7.3, and 7.4 show the confidence score distributions, represented by boxplots, for ORDREC, CPMF, CBPMF, LBD, and PRGAT models across the Amazon Movies TVs, Jester Joke, and MovieLens 1M datasets, respectively. Examining these distributions reveals varying behaviors across the models. ORDREC consistently exhibits a tight distribution. CPMF shows more variation in its confidence score range across datasets, but generally falls within a moderate to high confidence spectrum. CBPMF's confidence distribution also varies across datasets, appearing more spread out than ORDREC or LBD, yet still having a clear central tendency. LBD demonstrates an extremely low confidence for Movie Lens 1M, with scores clustered around zero. PRGAT also appears more spread, with a tendency toward higher confidence.

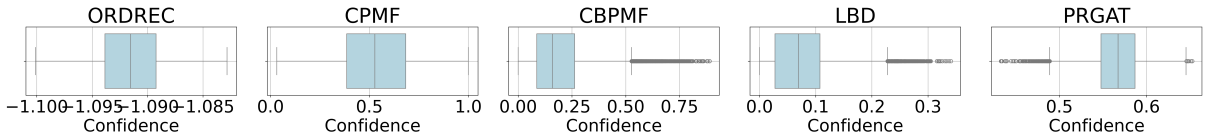

Figure 7.2: Distribution of confidence for each distribution-based model on the first test split of the Amazon Movies and TVs dataset.

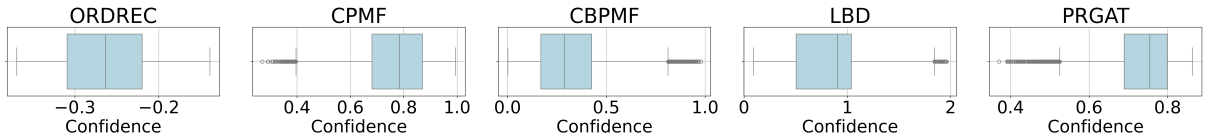

Figure 7.3: Distribution of confidence for each distribution-based model on the first test split of the Jester Joke dataset.

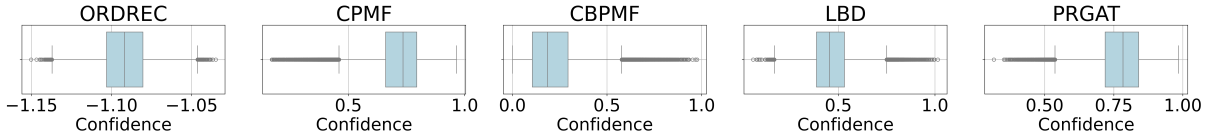

Figure 7.4: Distribution of confidence for each distribution-based model on the first test split of the Movie Lens 1M dataset.

Table 7.5 indicates that, among the distribution-based models, CPMF shows moderate negative correlations on Jester Joke and MovieLens 1M, but a positive correlation on

Table 7.5: Correlation between model confidence and prediction error across three datasets. Higher model confidence corresponds to lower prediction error.

Model	Amazon Movies and TVs	Jester Joke	Movie Lens 1M
ORDREC	-0.0003 ± 0.0061	0.0531 ± 0.0701	0.0189 ± 0.0055
CPMF	0.1675 ± 0.0854	-0.3296 ± 0.0985	**-0.3349 ± 0.0561**
CBPMF	-0.0179 ± 0.0067	-0.0426 ± 0.0248	-0.0128 ± 0.0035
LBD	0.0059 ± 0.0053	-0.1459 ± 0.0263	0.0145 ± 0.0098
PRGAT	**-0.1810 ± 0.0424**	**-0.3586 ± 0.0342**	-0.2615 ± 0.0057

Amazon Movies and TVs, suggesting inconsistent confidence calibration across datasets. ORDREC, CBPMF, and LBD exhibit correlations near zero, indicating little relationship between confidence and error. In contrast, the PRGAT model demonstrates consistently moderate negative correlations across all datasets. Despite this consistency, the correlation still leaves room for improvement.

The prior distribution-based models evaluated performed poorly in the datasets compared to the baselines and the proposal, PRGAT. Additionally, their performance likely depends on the dataset used, as no single model outperforms across all datasets. We can see that no model outperforms in all metrics in the Amazon Movies TVs (Table **??**), CPMF performs better in the Jester Joke (**??**), and LBD in the Movie Lens (**??**). Additionally, the correlation between confidence and error raises concerns, as also observed by Coscrado and Bridge (2023) in their study, which did not include CPMF and LBD. However, it's observable that only CPMF yields a promising confidence correlation with error for Jester Joke and Movie Lens 1M, while LBD only for the Jester Joke dataset.

The spike of confidence at some values suggests that the model is likely producing the exact value of confidence for most of its predictions. This makes sense when the model either performs poorly or very well in terms of accuracy. However, the performance of the prior distribution-based methods is moderate to poor compared to the MF baseline, which typically suggests a more dispersed confidence distribution; yet, this expected behavior is not observed. These results are more effectively discussed in the next section, which begins with a recap of the first finding.

7.6 DISCUSSION

Similar performances are expected across MF-based methods, regardless of the optimization strategy, as they share the same architecture for score prediction. However, the optimization method and goal affected some metrics. Additionally, the results indicated that optimizing only the negative log likelihood loss (NLL) does not yield better accuracy and ranking performance compared to using the MSE loss. Since we observed that the baseline MF outperformed all prior distribution-based models, MF-based performances were clearly harmed due to confidence integration. On the other hand, the results from PRGAT have demonstrated that the combination of NLL with MSE in the loss function, using stochastic gradient descent and a GAT-based architecture, yields strong results.

This combination improved the model's architecture performance. Since PRGAT is built upon DGAT and leverages confidence modeling from CPMF, we expect its performance to be lower than that of DGAT.

Through these conducted experiments, we can draw some reasons why the prior distribution-based models have the described results. The CPMF model offers several theoretical advantages, particularly regarding its probabilistic formulation. However, certain design aspects may hinder its empirical performance on real-world datasets, such as its optimization objective, regularization strategy, and model architecture. The model aims to minimize the NLL, yet its gradient often tends to vanish before achieving the same RMSE levels as standard MF. Furthermore, the applied regularization may further obscure the gradient direction, amplifying the challenges of optimizing an already weak loss signal.

The LBD model can be interpreted as a hybrid between OrdRec and CPMF, employing the Beta distribution. In theory, this formulation combines the strengths of both approaches while leveraging the flexibility of the Beta distribution to represent a wide range of shapes, including highly peaked distributions. Nevertheless, the resulting optimization problem becomes considerably more complex, making convergence difficult. For instance, this distribution flexibility can also be a disadvantage, such as the distribution assuming extreme shapes, specifically the U-shape ($\alpha < 1$ and $\beta < 1$) or J-shapes (one parameter < 1, the other > 1). These configurations cause the probability density function to approach infinity at one or both boundaries. This boundary behavior leads to catastrophic issues in the log-likelihood gradient, resulting in exploding gradients that are numerically unstable and prevent the SGD-based algorithm from performing the optimization effectively.

We observe a promising but large room for improvement of the correlation between confidence and error from the PRGAT. The proposed optimization goal for PRGAT may conflict after a point in the optimization process, preventing PRGAT from achieving the higher confidence correlation. For instance, this conflict can be caused by low precision in the evaluated confidence interval, defined as $P(r_{ui} = r|\Theta) = P(r_{ui} \leq r + \delta_r) - P(r_{ui} \leq r - \delta_r)$, where we may not have set a good value for δ_r. The authors of OrdRec may have recognized this issue by setting learned thresholds, but the method's confidence correlation is still not improved. This raises the necessity of exploring other modelings for $P(r_{ui} = r|\Theta)$. Another possible approach is to explore the addition of a calibration for those confidence estimations. For example, the work in (MOON et al., 2020) presents a solution to establish a relationship between confidence and error for neural network-based models, by setting an objective to adjust the confidence accordingly.

Overall, although the prior distribution-based models and the proposed approach are theoretically well-founded, none of them enforce a strong constraint in the loss function to ensure that the model's confidence is negatively correlated with prediction error. Among these, the experimental evaluation and the proposed method have some constraints and limitations.

7.7 LIMITATIONS

Despite the proposed method demonstrating strong performance in terms of accuracy and ranking metrics, its generalizability remains uncertain, as we have explored only three different datasets within distinct contexts each. Still, other datasets, especially those from diverse contexts, may exhibit significant variations in their existing patterns. Thus, the extension of this method to new datasets depends on conducting new experiments, making possible adjustments, and evaluating its applicability. Additionally, the proposed confidence modeling still requires improvements, considering that a perfect relationship with error would be desired: when high confidence, low error; when low confidence, up to high error. Finally, note that the proposed loss function formulation, as well as the prior-distribution-based methods, does not address the case of negative examples. Consequently, the model is trained only on positive examples and may perform poorly when encountering negative ones.

7.8 SUMMARY

This chapter presented the results of the proposed approach to enhancing recommender systems by integrating confidence into GAT and MF-based models. This section also describes the methodology used, the specific models employed, and the datasets utilized in the study. It then demonstrated the significant potential of the proposed solution for improving recommendation quality. Visual insights were also presented to analyze the quality of the model's confidence estimations. Finally, the chapter discussed the important limitations that remained. The solution focused on learn-to-rank is then presented in the next chapter.

Chapter 8

This experimental evaluation evaluates the "Gaussian Uncertainty-aware Learn-to-Rank" variation of the proposed solution. This chapter will evaluate the proposed approaches that integrate confidence into GAT and MF-based models, focusing on ranking metrics and the correlation between confidence and error.

EXPERIMENTAL EVALUATION 2: ASSESSING THE LISTWISE METHOD

This solution is focused on the pairwise learn-to-rank method, rather than rating prediction. We evaluate this second variation of our proposal following a similar strategy to the first evaluation. Therefore, we begin with the methodology, describe the dataset, define the evaluation metrics, and present the results. We then discuss the results and present the experimental evaluation, as well as the limitations of the proposed solution.

8.1 METHODOLOGY

We evaluate the integration of confidence over two models: the standard MF with bias and a deep GAT model, as described in Section 2 and Section 2.6. The embedding sizes of the models are set to 64 empirically. A Monte Carlo cross-validation strategy is implemented with 5 executions, where the first run utilizes a sorted dataset by timestamp, if available. Each execution uses a split of approximately 2/3 for training, 1/6 for validation during training, and 1/6 for testing.

The negative sampling number is set to 100 items per user. This number is set according to the available computational resources and time. Since this number was kept consistent across all methods, it does not affect the fairness of the comparison or the relative performance evaluation among models. The additional negative items for a specific user are defined as the set of items that have been interacted with by at least one user but not rated by the target user. In the evaluation, this constructed subset of candidates is ordered by the ranking scores produced by the models. We evaluate the models in the evaluation and test subsets; however, we bring only the performance results from the test subset.

All the aforementioned models were trained using the Adam optimization algorithm with a learning rate of 0.001. The methods were implemented in PyTorch using the

Python programming language, where the default learning rate for Adam is 0.001. Early stopping was applied as a regularization strategy across all training procedures, with a patience threshold of 5 epochs without improvement in the evaluation error. A patience of 5 is set based on empirical observations. The next section describes the datasets used in this experimental evaluation.

8.2 DATASETS

The experiments utilize three publicly available datasets: Amazon Beauty, Jester Joke, Movie Lens, and Rotten Tomatoes. These datasets were selected based on the related works to cover distinct recommendation scenarios. They are all tabular, containing user ID, item ID, the explicit feedback rating, and timestamp columns.

The MovieLens 1M dataset[1] is a widely used benchmark in recommender systems research, especially for evaluating collaborative filtering algorithms in the context of movie preferences. It contains 1,000,209 explicit ratings ranging from 1 to 5, provided by 6,040 users over 3,416 movies. Due to the large number of possible user-item pairs, the dataset exhibits a high sparsity level of 0.9553.

The Jester Joke dataset[2] focuses on humor preferences and differs from typical recommendation datasets in its continuous rating scale from -10 to 10, with a finer-grained preference description. It includes 1,800,475 ratings from 24,982 users over 99 jokes. It has an extremely sparse matrix (sparsity = 0.9999).

The Amazon Beauty dataset[3], derived from Amazon's product reviews, represents a real-world e-commerce domain with implicit and explicit signals of user preferences. It contains 2,023,070 interactions, ranging from 1 to 5 stars, across 67,134 beauty-related products, provided by 840,790 users. To reduce noise and emphasize meaningful feedback, items with fewer than ten interactions are excluded. This dataset features a relatively lower sparsity level of 0.2720.

The Rotten Tomatoes dataset[4] Compiled from the Rotten Tomatoes movie review platform, this dataset captures user ratings and critiques in the entertainment domain. It comprises 553,207 interactions between 2,614 users and 17,484 movies, reflecting both individual preferences and collective opinions. Similar to other recommendation benchmarks, items with a limited number of interactions are filtered to ensure reliable evaluation. This dataset exhibits a high sparsity level of 0.9878, characteristic of real-world rating scenarios where users interact with only a small fraction of available items.

All of these datasets underwent a preprocessing stage, which included removing users and items with fewer than five interactions, as well as excluding any users or items in the test or evaluation sets that did not appear in the training set. Table 8.1 summarizes the datasets after these preprocessing steps.

We perform an evaluation of models' performance regarding ranking metrics and the

[1] https://grouplens.org/datasets/MovieLens/1m/

[2] Available at <https://eigentaste.berkeley.edu/dataset/>

[3] Available at: <https://cseweb.ucsd.edu/~jmcauley/datasets/amazon/links.html>

[4] Available at: <https://www.kaggle.com/datasets/stefanoleone992/rotten-tomatoes-movies-and-critic-reviews-dataset>

Table 8.1: Summary of the datasets adopted. The n_u and n_i are the number of users and items, respectively. The n_{inter} is the number of interactions. The ratios for training, evaluation, and test set are $train_r$, $eval_r$, and $test_r$

dataset	n_u	n_i	n_{inter}	sparsity	$train_r$	$eval_r$	$test_r$
Amazon Beauty	40030	39086	332358	0.999788	0.726163	0.13692	0.13692
Jester Joke	24982	99	1800475	0.272011	0.799932	0.10003	0.10003
Movie Lens 1M	6040	3416	999611	0.951552	0.747815	0.12609	0.12609
Rotten Tomotoes	2612	13631	542889	0.984752	0.753664	0.12317	0.12317

Table 8.2: Model's performance over the test split of the Amazon Beauty dataset.

metric	UA-GAT	UA-MF	DGAT	MF
NDCG@10	0.5128 ± 0.0183	0.6338 ± 0.0142	0.5301 ± 0.0157	**0.6378** ± 0.0081
MAP@10	0.4894 ± 0.0239	**0.6616** ± 0.0139	0.5111 ± 0.0256	0.6022 ± 0.014
Recall@10	0.5045 ± 0.035	**0.7841** ± 0.0214	0.5236 ± 0.0426	0.6105 ± 0.0199
MRR@10	0.7752 ± 0.0397	0.8469 ± 0.0114	0.7705 ± 0.0175	**0.8583** ± 0.0289
NDCG@3	0.5933 ± 0.0353	0.7277 ± 0.0164	0.5916 ± 0.0184	**0.7291** ± 0.0108
MAP@3	0.5803 ± 0.0362	**0.7424** ± 0.0161	0.5833 ± 0.0227	0.7197 ± 0.0093
Recall@3	0.5833 ± 0.0379	**0.7939** ± 0.0181	0.5818 ± 0.0223	0.7197 ± 0.0093
MRR@3	0.7583 ± 0.0426	0.8333 ± 0.011	0.7515 ± 0.0198	**0.8485** ± 0.0271

confidence-error relationship for each of those datasets.

8.3 EVALUATION METRICS

The metrics utilized for evaluating the ranking performance of the models are NDCG@10, NDCG@3, mean reciprocal rank (MRR)@10, MRR@3, MAP@3, MAP@10, Recall@10, and Recall@3. For confidence-error correlation, we use Pearson correlation. These metrics are applied to each model in each dataset, allowing us to assess its results.

8.4 RESULTS

From previous works that integrate uncertainty into a model, the performance in ranking metrics tends to decrease. However, Table 8.2 reports an unexpected result: UA-MF outperforms the baselines in four metrics, including MAP@10 (0.6616), Recall@10 (0.7841), MAP@3 (0.7424), and Recall@3 (0.7939). Additionally, UA-MF maintains a highly close performance with the baseline MF, which slightly outperforms other models in NDCG@10 (0.6378), MRR@10 (0.8583), NDCG@3 (0.7291), and MRR@3 (0.8485). This indicates strong ranking consistency with and without confidence integration. In contrast, UA-GAT and DGAT perform notably worse across all evaluation metrics, with scores consistently below those of the other two methods. This also indicates that GAT-based models perform poorly in the Amazon Beauty dataset. The comparable performance of all mod-

Table 8.3: Model's performance over the test split of the Movie Lens 1M dataset.

metric	UA-GAT	UA-MF	DGAT	MF
NDCG@10	**0.9296** ± 0.0272	0.7563 ± 0.0879	0.8647 ± 0.1854	0.7635 ± 0.0919
MAP@10	**0.9192** ± 0.016	0.7518 ± 0.0805	0.8613 ± 0.1703	0.7362 ± 0.0828
Recall@10	**0.9136** ± 0.0223	0.6358 ± 0.045	0.8573 ± 0.1673	0.7362 ± 0.0828
MRR@10	**0.9986** ± 0.0029	0.9089 ± 0.0743	0.9315 ± 0.1526	0.9139 ± 0.078
NDCG@3	**0.9934** ± 0.0078	0.8212 ± 0.1054	0.8922 ± 0.236	0.8301 ± 0.1143
MAP@3	**0.993** ± 0.0067	0.8216 ± 0.1011	0.8916 ± 0.2367	0.8244 ± 0.1118
Recall@3	**0.9928** ± 0.0072	0.7677 ± 0.0883	0.8915 ± 0.2366	0.8244 ± 0.1118
MRR@3	**0.9985** ± 0.0031	0.9049 ± 0.0807	0.9246 ± 0.168	0.9098 ± 0.0856

Table 8.4: Metrics of the models in the test split of Jester Joke.

metric	UA-GAT	UA-MF	DGAT	MF
NDCG@10	0.3588 ± 0.0673	0.5243 ± 0.008	0.3494 ± 0.0284	**0.5402** ± 0.0039
MAP@10	0.3529 ± 0.0582	0.5148 ± 0.0065	0.3534 ± 0.0237	**0.5324** ± 0.0045
Recall@10	**0.6471** ± 0.0582	0.4852 ± 0.0065	0.6409 ± 0.0327	0.4689 ± 0.0037
MRR@10	0.5495 ± 0.0876	0.7132 ± 0.0361	0.5353 ± 0.054	**0.7137** ± 0.0223
NDCG@3	0.3757 ± 0.1039	0.5502 ± 0.0282	0.3558 ± 0.0595	**0.5621** ± 0.0095
MAP@3	0.3795 ± 0.1122	0.5528 ± 0.0233	0.3662 ± 0.0673	**0.5721** ± 0.0087
Recall@3	**0.6205** ± 0.1122	0.4472 ± 0.0233	0.6113 ± 0.109	0.4277 ± 0.01
MRR@3	0.5092 ± 0.1136	0.6903 ± 0.0416	0.4887 ± 0.0682	**0.6908** ± 0.0287

Table 8.5: Metrics of the models in the test split of Rotten Tomatoes.

metric	UA-GAT	UA-MF	DGAT	MF
NDCG@10	**0.7848** ± 0.0337	0.7287 ± 0.0033	0.7615 ± 0.0267	0.7391 ± 0.0097
MAP@10	**0.7939** ± 0.0487	0.7245 ± 0.003	0.7469 ± 0.0422	0.7164 ± 0.0072
Recall@10	**0.7759** ± 0.0167	0.6777 ± 0.0073	0.7692 ± 0.0188	0.7165 ± 0.0072
MRR@10	**0.9618** ± 0.0148	0.8861 ± 0.012	0.9458 ± 0.0109	0.8889 ± 0.01
NDCG@3	**0.9036** ± 0.0261	0.7914 ± 0.0143	0.881 ± 0.0225	0.7947 ± 0.0174
MAP@3	**0.9012** ± 0.0291	0.7968 ± 0.015	0.8744 ± 0.0248	0.7906 ± 0.0176
Recall@3	**0.8916** ± 0.0125	0.7637 ± 0.0072	0.8853 ± 0.0187	0.7906 ± 0.0176
MRR@3	**0.9585** ± 0.0154	0.8793 ± 0.0134	0.9416 ± 0.0116	0.8824 ± 0.0121

Table 8.6: Confidence correlation with BPR error.

Dataset	UA-GAT	UA-MF
Amazon Beauty	-0.7309 ± 0.0302	**-0.8227 ± 0.0215**
Jester Joke	-0.2638 ± 0.0821	**-0.6416 ± 0.0145**
Movie Lens 1M	**-0.7542 ± 0.0754**	-0.5273 ± 0.0243
Rotten Tomatoes	**-0.7375 ± 0.1178**	-0.6385 ± 0.0096

els with and without confidence integration indicates that incorporating confidence does not degrade their effectiveness in this dataset.

Table 8.3 shows that UA-GAT clearly dominates across all evaluation metrics. These metric performances indicate near-perfect retrieval quality, particularly at the top ranks, where early precision and recall are consistently close to 1. The other models lag considerably behind: DGAT shows competitive but lower results, while UA-MF and MF trail further with substantially weaker performance across all metrics. The results suggest that the graph-based, especially the UA-GAT variant, is well-suited for the Movie Lens 1M dataset, outperforming both matrix factorization and the DGAT variants by a wide margin.

Table 8.4 shows a more competitive landscape compared to other datasets. MF achieves the best results in NDCG@10, MAP@10, MRR@10, NDCG@3, MAP@3, and MRR@3, indicating strong ranking consistency. Meanwhile, UA-MF delivers the second-best scores across all these metrics, with a margin of superiority over the other models. Interestingly, in terms of recall, UA-GAT and DGAT perform better, with the former achieving the highest score. This suggests that while MF-based models excel in precision-oriented measures, GAT-based variants show an advantage in retrieving a broader set of relevant items.

The Table 8.5 shows that the UA-GAT model consistently outperformed the other models across all metrics at both cutoffs, 10 and 3 for the Movie Lens 1M dataset. The DGAT model ranked second. The UA-MF and MF models generally exhibited lower performance across all metrics. Similarly, the results found using the Movie Lens dataset have shown GAT-based models to be more suitable, especially with confidence integration.

Overall, regarding ranking performance, integrating confidence into the model does not degrade its effectiveness. On the contrary, the proposed confidence-aware variants outperform the baseline in the movie domain datasets, MovieLens 1M and Rotten Tomatoes. In the remaining datasets, where the proposed models do not achieve the best overall results, they still either outperform the baseline in specific metrics or remain a close second, demonstrating consistent and competitive performance across different contexts.

The Table 8.6 reported Pearson correlation coefficients, with negative values indicating that higher confidence is associated with lower error, and standard deviations are provided to show variability. Values close to -1 are desired. The results show a strong negative correlation across all datasets, indicating that higher confidence values are generally associated with lower prediction errors. This confirms that the confidence estimation

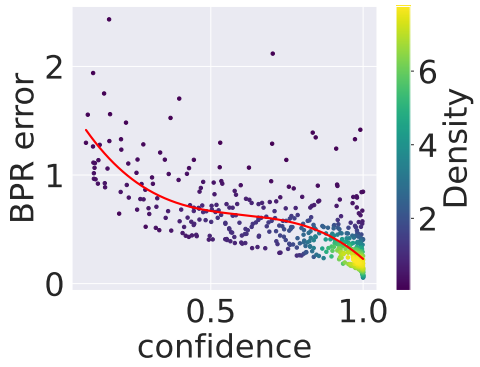

Figure 8.1: Scatter plot of confidence versus BPR error for the Amazon Beauty dataset using the UA-GAT model. The fitted curve found is $BPR_{error}(conf) = -4.627conf^3 + 8.389conf^2 - 5.370conf + 1.835$.

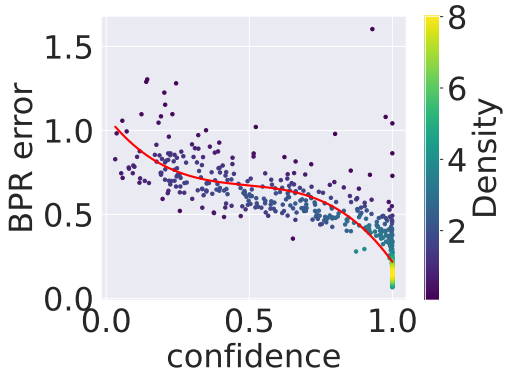

Figure 8.2: Scatter plot of confidence versus BPR error for the Amazon Beauty dataset using the UA-MF model. The fitted curve found is $BPR_{error}(conf) = -2.806conf^3 + 4.128conf^2 - 2.193conf + 1.087$

effectively reflects the model's uncertainty. Specifically, UA-MF exhibits the strongest correlation in the Amazon Beauty and Jester Joke datasets, whereas UA-GAT achieves higher correlations in the Movie Lens 1M and Rotten Tomatoes datasets. These findings suggest that the confidence learned by both models is meaningfully aligned with their predictive reliability, with each architecture performing better in different data domains. Another noteworthy observation is that stronger negative correlations are observed in datasets where each architecture performs best. For instance, UA-GAT achieves superior results on MovieLens 1M and Rotten Tomatoes, both within the movie domain, and correspondingly exhibits higher negative correlations in these datasets. Conversely, the MF-based models perform better on Amazon-Beauty and Jester-Joke, where UA-MF also shows stronger negative correlations. This consistency reinforces the relationship between the reliability of the confidence estimates and the overall performance of each architecture in its most suitable data domain.

The scatter plot presented in Figures 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8 depicts the

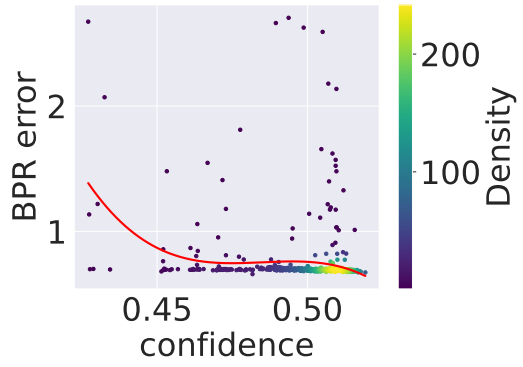

Figure 8.3: Scatter plot of confidence versus BPR error for the Jester Joke dataset using the UA-GAT model. The fitted curve found is $BPR_{error}(conf) = -3594.773conf^3 + 5229.297conf^2 - 2534.600conf + 410.080$

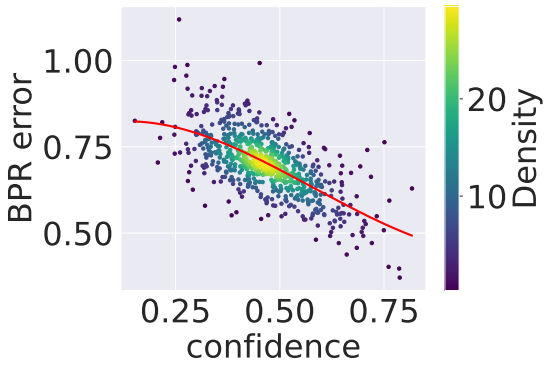

Figure 8.4: Scatter plot of confidence versus BPR error for the Jester Joke using the UA-MF model. The fitted curve found is $BPR_{error}(conf) = 1.108conf^3 - 1.927conf^2 + 0.468conf + 0.793$

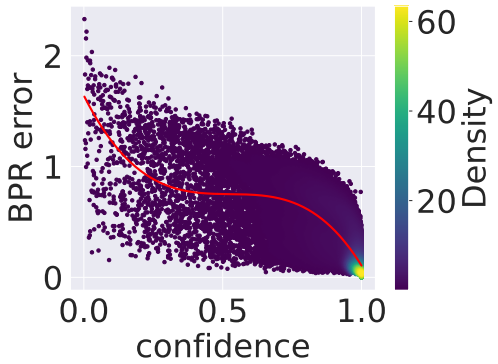

Figure 8.5: Scatter plot of confidence versus BPR error for the Movie Lens 1M dataset using the UA-GAT model. The fitted curve found is $BPR_{error}(conf) = -5.967conf^3 + 9.438conf^2 - 5.006conf + 1.641$

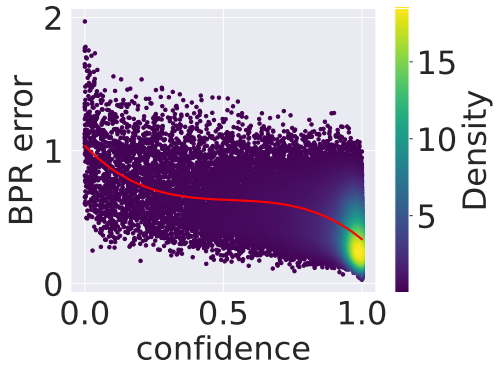

Figure 8.6: Scatter plot of confidence versus BPR error for the Movie Lens 1M dataset using the UA-MF model. The fitted curve found is $BPR_{error}(conf) = -2.448conf^3 + 3.889conf^2 - 2.144conf + 1.036$

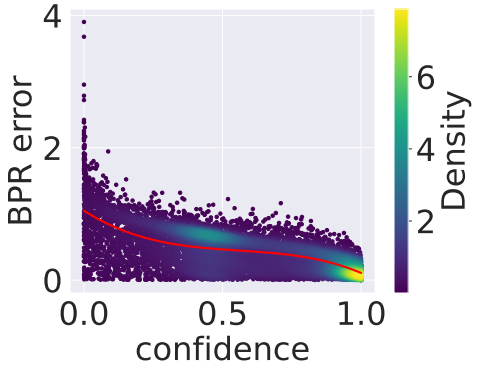

Figure 8.7: Scatter plot of confidence versus BPR error for the Rotten Tomatoes dataset using the UA-GAT model. The fitted curve found is $BPR_{error}(conf) = -2.557conf^3 + 4.297conf^2 - 2.681conf + 1.049$

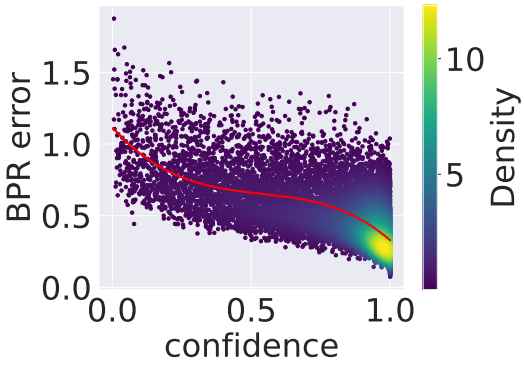

Figure 8.8: Scatter plot of confidence versus BPR error for the Rotten Tomatoes dataset using the UA-MF model. The fitted curve found is $BPR_{error}(conf) = -2.255 \cdot conf^3 + 3.628conf^2 - 2.161conf + 1.116$.

relationship between confidence and BPR error. The x-axis represents the confidence scores, ranging from 0.0 to 1.0, while the y-axis corresponds to the BPR error. The distribution of points shows a clear downward trend: as confidence increases, the BPR error tends to decrease. This inverse relationship indicates that higher confidence scores are associated with lower prediction errors. A cubic fitted curve overlays the scatter plot,

$$\mathrm{BPR}_{error}(conf) = a \cdot \mathrm{conf}^3 + b \cdot \mathrm{conf}^2 + c \cdot \mathrm{conf} + d\,,$$

evidencing this negative correlation. Additionally, the marginal density plots along the axes reveal that confidence scores are more concentrated toward higher values (closer to 1.0). In contrast, BPR error values cluster at lower ranges.

8.5 DISCUSSION

Compared to the original formulation of the BPR loss, the main advantage of the proposed loss function, while maintaining the objective, is its use of the cumulative distribution function (CDF) of the Normal distribution instead of the logistic sigmoid function. Although both functions share a similar shape, the CDF formulation is more appropriate in this context, as it naturally incorporates both the mean and variance when modeling the probability under a Gaussian assumption. This design enables the model to simultaneously estimate the likelihood that an observed item receives a higher score than a non-observed one and quantify the associated uncertainty (i.e., the predicted standard deviation), without requiring any additional optimization term.

The stronger performance observed in the movie-domain datasets for both baseline and proposed models indicates that the architecture is better suited for movie-oriented data. Since the models treat graph connections (user–item interactions) independently and without considering temporal or sequential order, their effectiveness diminishes in scenarios where interactions are interdependent or occur in a specific order. The high sparsity is another factor that has contributed to the smaller performance of Jester Joke. This limitation is directly reflected in the models' uncertainty estimates. For instance, when comparing the performance in the movie-domain datasets to that in Jester Joke, both the predictive performance and the strength of the confidence–error correlation decrease noticeably. This behavior is expected and suggests that the models' uncertainty estimates are formulated, reflecting the models' reliability. Consequently, the dataset-specific alignment between model performance and confidence correlation reinforces the validity of the uncertainty modeling. This alignment is further visually confirmed by the fitted cubic relationship between confidence and BPR error, which consistently shows that higher confidence corresponds to lower prediction error and a poor shape for the variant with poor performance in the dataset.

8.6 LIMITATIONS

The results discussed are constrained to the datasets and models employed in this study. As observed, performance and calibration vary depending on both the dataset and the model architecture. Therefore, we can conclude that the proposed method produces a

dataset-dependent calibration. Although the correlation between confidence and error effectively reflects model reliability, this calibration tends to degrade when the model fails to generalize well to a particular dataset. Such degradation may stem from dataset characteristics, including sparsity, noise, or the complexity of underlying patterns, as well as from the model architecture itself. A model may inherently capture specific patterns that are absent from certain datasets, while failing to learn those that are actually present. Additionally, this study did not investigate how confidence-aware decision strategies might impact ranking performance, particularly in scenarios where the model underperforms on a specific dataset.

8.7 SUMMARY

This chapter presented our method for integrating confidence estimation into embedding-based recommendation models, specifically matrix factorization and graph attention network architectures. The results demonstrated that incorporating confidence preserves ranking performance and does not negatively impact baseline effectiveness. In several domains—particularly those aligned with each model's inherent strengths—the confidence-aware variants achieved notable performance gains. We further showed that the learned confidence values serve as reliable indicators of predictive certainty, exhibiting strong negative correlations with BPR error. Our analysis highlighted clear architecture–dataset affinities, revealing when UA-MF or UA-GAT benefits most from confidence integration. Overall, the findings confirm the robustness, interpretability, and practical value of confidence-aware modeling across diverse datasets and ranking metrics. The next chapter concludes the work, highlights the dissertation's contributions, and presents future research directions.

Chapter 9

This chapter presents the conclusions of the work, states the contributions obtained by this study, answers the research question, and presents the future works.

CONCLUSION

We presented two proposals and evaluated them. Specifically, this work presented a benchmark of the previous methods for confidence estimation in RecSys, which are OrdRec, CPMF, CBPMF, and LBD. We have compared these methods against each other and the standard MF with bias, evaluating their accuracy, ranking performance, and confidence quality. Then, we proposed a new method for integrating confidence into models for rating prediction, based on previous confidence methods, closing the first proposal. Additionally, this work introduced an uncertainty-aware learning-to-rank architecture and validated it through experiments on four benchmark datasets: Amazon Beauty, Jester Joke, Movie Lens 1M, and Rotten Tomatoes.

For the first experimental evaluation, we analyzed the performance metrics, confidence distribution, and relationship with the absolute error of the prior methods. This work then explored two models: DGAT and PRGAT, where the first is a deep graph neural network, and the second is a confidence integration model built on top of DGAT. Experiments showed that no prior distribution-based method consistently outperforms all metrics across all datasets. Thus, previous distribution-based methods are dependent on the dataset, as discussed in this dissertation, regarding the metrics presented. Among the prior distribution-based models, no single method generalizes well across datasets; CPMF performs best on Jester Joke, while LBD performs best on MovieLens 1M, and none achieve consistent ranking superiority. Their probabilistic formulations, although theoretically appealing, introduce significant optimization challenges that limit empirical performance. On the other hand, the proposed method, which is also distribution-based, has demonstrated strong performance across all metrics and datasets used, outperforming all previous distribution-based methods. Although it does not consistently outperform models without confidence integration on all datasets, the proposed method is a strong option due to its consistency across various metrics and datasets in terms of accuracy, ranking, and correlation with confidence and error. Specifically, among the prior distribution-based methods, only the CPMF method has presented reasonable confidence

scores, particularly in terms of their distribution compared to their accuracy performance and their correlation with error. The proposed method leverages the CPMF confidence approach and presents comparable confidence quality when evaluated in terms of overall aspects. However, as discussed previously, both the prior and proposed methods still require improvements: the confidence scores still need a better relationship with the model's error.

For the second experimental evaluation, the results demonstrate that the proposed approach not only preserves ranking performance but can also enhance it in certain datasets and model variants. Furthermore, the observed correlation between confidence estimates and prediction error indicates that the proposed method provides a well-calibrated measure of model reliability, particularly in datasets where the model effectively captures underlying patterns. Overall, the proposed framework proves to be well-suited for learning-to-rank tasks and is potentially generalizable to other model architectures and ranking loss formulations, such as RankNet. Methodologically, employing the Gaussian CDF within the BPR loss proved effective for modeling both mean and variance, which offers a theoretically grounded way to integrate uncertainty without additional optimization terms. The results reveal that the proposed method's calibration and performance are domain-dependent, performing particularly well in entertainment-oriented datasets, while facing challenges in domains with more complex or sequential interaction patterns. We conclude that this behavior is highly correlated with the chosen model's learning capabilities for the dataset, as baselines and proposals exhibit similar behavior, which we experimented with using MF and GAT. This highlights that architectural design has a significant influence on both predictive performance and uncertainty calibration.

To highlight the specific advances produced in each research direction, the contributions of this thesis are summarized below in two sections. The first section covers the contributions obtained in the rating-prediction setting, and the second examines the contributions resulting from the learn-to-rank study.

9.1 CONTRIBUTIONS

The first part of this dissertation brings the following contributions:

- Comprehensive empirical evaluation of prior distribution-based confidence models. We demonstrate that no prior distribution-based method consistently performs well across datasets or metrics.
- Identification of optimization and stability issues in existing probabilistic MF models: a diagnostic explanation of why these models often underperform and provide actionable insight into their optimization difficulties.
- Demonstration that confidence integration harms MF-based architectures under NLL optimization alone.
- Validation that integrating probabilistic modeling into a GAT architecture (PRGAT) produces equivalent or superior recommendation quality and more meaningful confidence estimates than both MF-based and prior distribution-based methods.

- Characterization of confidence score distributions across models.
- A set of theoretically grounded hypotheses about where further improvements in confidence calibration can be made.

The second part of this dissertation brings the following contributions:

- A unified uncertainty-aware learn-to-rank formulation using a Gaussian CDF, which generalizes across BPR and RankNet methods.
- Confidence-aware variants that preserve or improve ranking performance across four datasets and two model families.
- Empirical evidence that uncertainty reflects predictive reliability with strong negative Pearson correlations between confidence and BPR error. The cubic fitted curves across datasets consistently show that higher confidence corresponds to lower prediction error.
- Clear architecture–dataset alignment revealed through uncertainty analysis. While MF-based variants outperform on datasets such as Amazon Beauty and Jester Joke, and correspondingly show stronger confidence–error correlations in these domains. GAT-based models, especially UA-GAT, dominate the movie datasets, again accompanied by stronger correlations.
- Robustness insights through dataset difficulty characterization. Uncertainty magnitudes and correlations reflect the properties of the dataset, such as sparsity, interaction independence, and domain complexity.
- A cross-domain evaluation demonstrating general applicability, the method is validated across four heterogeneous datasets (beauty products, jokes, movies).

9.2 FUTURE WORKS

To address rating prediction confidence issues, future work should pursue the optimization directions proposed in this study, aiming to enhance the model's performance. Additionally, investigating new datasets across diverse contexts is recommended to evaluate the generalizability of the proposed approach using the selected metrics. As noted, while a dataset–model alignment effect exists in recommender systems, the criteria that determine architectural suitability remain poorly understood. As future work, we plan an extensive study to identify the factors that influence the suitability of a machine learning architecture for a given dataset. Furthermore, proposing new confidence estimation approaches and assessing their estimates may yield valuable insights and guide future improvements. Finally, but not only, investigate calibration for the confidence estimations.

In future work on learn-to-rank confidence, we can explore applications of the confidence for filtering, weighting, or explaining recommendations. Future work could explore confidence integrated in sequential models to improve robustness across domains. As future work, we also plan to explore diversity and serendipity, for example, by replacing low-confidence recommended items with novel and potentially surprising ones.

9.3 PUBLICATIONS

The following is the list of articles that the authors are publishing during this research.

1. **Towards Prediction Error Reduction in Matrix Factorization Recommenders Using Semantic Similarity Metrics**

   Status: *Published*

   Conference: *Americas Conference on Information Systems (AMCIS) 2025*

   Abstract: Recommender systems leverage historical data to provide personalized suggestions, often using Matrix Factorization (MF) techniques like SVD. However, data sparsity remains a key challenge to high precision. Accurate recommendation predictions are crucial for platform success, as significant errors can severely impact user experience. When recommendations consistently fail to meet expectations, users lose trust in the system, resulting in decreased engagement and ultimately, platform abandonment. Our research addresses this critical challenge by developing a novel hybrid recommendation approach that reduces prediction error. By combining matrix factorization with DBpedia-derived semantic similarities, we achieve more precise personalization that better aligns with user preferences. Our approach minimizes sparsity by inferring missing ratings for semantically similar items. We therefore propose a hybrid framework combining similarity-based imputation with MF, merging content-based and collaborative filtering advantages, targeting improving prediction rates through strategic rating pre-filling.

2. **A Gated Review Attention Framework for Topics in Graph-Based Recommenders**

   Status: *Published*

   Conference: *Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia) 2025*

   Abstract: Recommender systems significantly reduce information overload by curating personalized content on digital platforms, thereby enhancing user experience. Traditional models often rely on sparse rating data, overlooking the rich semantic signals embedded in user reviews. To address this, we propose the Gated Review Attention Framework for Topics, a novel graph-based recommender system that integrates review-derived topic information with user-item interactions. Our approach leverages BERTopic to extract interpretable semantic topics from textual reviews and then embeds them into a graph attention network architecture. A gating mechanism dynamically regulates the influence of these topic representations relative to latent user and item embeddings, enabling adaptive feature fusion. We evaluate GRAFT on three benchmark datasets: Amazon Movies and TV, IMDb, and Rotten Tomatoes. Comparing it against classical and neural baselines, including SVD, DeepCoNN, and KANN. Experimental results demonstrate that GRAFT consistently achieves the lowest RMSE across all datasets, indicating superior rating prediction accuracy. Although traditional models perform better on ranking

metrics, GRAFT achieves superior accuracy (lower RMSE), and our qualitative analysis demonstrates more substantial semantic alignment.

3. **Software Development Using a Multi-agent Approach**

   Status: *Published*

   Conference: *Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações (WESAAC) 2025*

   Abstract: Software development involves complexities in requirements analysis, code generation, and continuous validation. This work presents a multi-agent system (MAS) designed to enhance the software engineering process by integrating Large Language Model (LLM) capabilities with human oversight. The MAS comprises five specialized agents that communicate through a shared message broker and maintain context in a persistent knowledge base. These agents autonomously handle tasks such as architectural planning, artifact generation, code review, and operational management, while human experts intervene to ensure accuracy and strategic alignment. The findings suggest that LLMs can significantly impact software engineering, providing tools and techniques that automate tasks, enhance code generation, and potentially improve the overall development process.

4. **Exploiting Surrogate Submodular and Cost-Effective Lazy Forward Algorithms for Calibrated Recommendations**

   Status: *Published*

   Conference: *40º Simpósio Brasileiro de Banco de Dados (SBBD) 2025*

   Abstract: Recommendation systems are tools to suggest items that might interest users. These systems rely on the user's preference history to generate a list of suggestions most similar to items in the user's history, aiming for better accuracy and minimal error. Pursuing higher accuracy can lead to side effects such as overspecialization, reduced diversity, and imbalances in categories, genres, or niches. Thus, this work explores algorithms for selecting the items that form a calibrated recommendation list. The hypothesis is that calibration can positively contribute to more accurate and fair recommendations based on user preferences. We introduce a variation of an existing algorithm that performs better than the baselines in a commonly used dataset, as measured by two different divergence metrics.

5. **HESTIA: A Home Environment Simulator Targeting Inhabitant Activities**

   Status: *Submitted*

   Journal: *IEEE Internet of Things Journal*

   Abstract: The diverse range of Internet of Things (IoT) devices available today offers a potential for home automation and smart home (SH) development, whether partial or complete. However, for research in this area to advance, it is essential to overcome the scarcity of data. In the context of data collection in SHs, the

main challenges include device compatibility and proprietary restrictions, as well as privacy and security concerns, and the inherent complexity of the environment's behavior. Furthermore, freely available datasets often present limitations, such as incomplete annotations, restricted device usage, and insufficient spatial and temporal coverage. To address this limitation, we developed a novel SH simulator that supports multi-state devices and captures detailed information on device usage and inhabitant activities, including metrics like lamp brightness and air conditioning settings (temperature, mode, etc.). In this work, we propose a home environment simulator targeting inhabitant activities (HESTIA), a simulator designed to support researchers in SHs and related fields by generating datasets that capture the activity patterns of individuals or groups in simulated environments. HESTIA generates synthetic data to address inconsistencies in device communication resulting from manufacturer-specific differences, thereby reducing the need for time-consuming and costly real-world data collection. This provides rich, dynamic information that can be tailored to specific user profiles. To validate the simulator, we use two datasets: one from the CASAS group and another collected from a real-world environment. For each dataset, we generate an equivalent synthetic version with HESTIA and compare the distributions using statistical tests. The results show that the simulator accurately replicates real-world environments, even for devices with inconsistent status changes. Additionally, we evaluated the computational complexity of this simulator in terms of time.

6. **Review and Experimental Evaluation of Machine Learning Methods for Lie Detection Using Microexpressions**

   Status: *Submitted*

   Journal: *International Journal of Computer Applications in Technology*

   Abstract: Facial analysis enables the recognition of emotional states and deceptive behavior, particularly through the detection of microexpressions, which are reliable indicators of concealed emotions. In this work, we conducted a scoping review of machine learning–based microexpression analysis methods for lie detection. The review included studies from IEEE, ACM, and Google Scholar, and we extracted information on model families, feature-extraction tools, classification methods, and evaluation strategies. From the literature scope review, we observed that decision tree-based classifiers combined with the OpenFace feature extractor are among the most frequently reported and effective approaches. To complement the review, we identified five representative algorithm groups and conducted an experimental evaluation on a common dataset validation strategy. Our results show that OpenFace, combined with a K-nearest neighbors classifier, achieved the highest accuracy. Finally, we provide insights into the observed findings and suggest directions for advancing lie detection methods, particularly in the context of deep learning.

7. **Multi-label Incremental Learning for State Prediction of Actuator Devices in Smart Homes**

Status: *Submitted*

Journal: *IEEE Internet of Things Journal*

Abstract: The main aspects considered in smart homes (SHs) include activity classification, routine prediction, energy consumption reduction, and actuator-state prediction. SHs require highly responsive behavior whenever state changes are detected. In this context, the key factors are both the response time and the quality of state prediction. However, many current approaches rely on deep learning models that demand substantial computational power and often require architectural changes whenever new device attributes are introduced. Moreover, the literature has predominantly focused on human activity prediction rather than device state prediction. In this work, we focus on actuator-state prediction and propose a flexible solution based on incremental learning, dynamic time windows, and data streaming, designed to accommodate frequent changes in user and device data structures independently of the underlying activity-recognition method. Using four datasets, one from CASAS, two generated by simulation, and one collected from a volunteer's SH, we evaluated the models in terms of Micro-F1 and per-instance inference time. The results show that, even in the most complex scenario, inference latencies for all algorithms remain in the millisecond range, making them practically negligible for real-time smart-home operation. Regarding predictive performance, adaptive random forest (ARF), k-nearest neighbors classifier (KNN), and extremely fast decision tree (EFDT) achieved the highest Micro-F1 scores across the different configurations, with ARF and EFDT offering an especially favorable balance between accuracy and responsiveness for deployment in real-world SH environments

8. **Understanding Distribution Structure on Calibrated Recommendation Systems**

Status: *Submitted*

Journal: *IEEE Transactions on Pattern Analysis and Machine Intelligence*

Abstract: Traditional recommender systems aim to generate a recommendation list comprising the most relevant or similar items to the user's profile. These approaches can create recommendation lists that omit item genres from the less prominent areas of a user's profile, thereby undermining the user's experience. To address this issue, the calibrated recommendation system ensures that less representative areas are included in the recommended list. The calibrated context works with three distributions. The first is from the user's profile, the second is from the candidate items, and the last is from the recommendation list. These distributions are G-dimensional, where G is the total number of genres in the system. This high dimensionality requires a different evaluation method, considering that traditional recommenders operate in a one-dimensional data space. In this sense, we implement fifteen models that help to understand how these distributions are structured. We evaluate the users' patterns in three datasets from the movie domain. The results indicate that the models of outlier detection provide a better understanding of the structures. The calibrated system creates recommendation lists that act

similarly to traditional recommendation lists, allowing users to change their groups of preferences to the same degree.

9. **Exploiting Distribution-Based Confidence Integration in Graph Neural Network Recommenders**

   Status: *Submitted*

   Journal: *Applied Intelligence (Springer)*

   Abstract: Recommender systems help users navigate information-rich environments by delivering personalized content. While model-based collaborative filtering approaches, such as matrix factorization (MF) and graph neural networks (GNN), are widely adopted, the inherent uncertainty in user preferences and sparse data can lead to unreliable predictions. Confidence estimation has emerged as a strategy to quantify prediction reliability; however, its integration remains unexplored in GNN-based models, and prior methods often compromise accuracy or suffer from convergence issues. This study benchmarks four prominent confidence-aware models—OrdRec, Confidence-aware Probabilistic Matrix Factorization, Confidence-aware Bayesian Probabilistic Matrix Factorization, and Lightweight Beta Distribution — across three public datasets: Amazon Movies and TV, Jester Joke, and MovieLens. We evaluate these models in terms of rating accuracy (root mean squared error), ranking quality (normalized discounted cumulative gain, mean average precision), and the quality of their confidence estimates. In addition, we propose a novel confidence-integrated model based on a deep graph attention network architecture. Experimental results reveal that while distribution-based confidence methods are highly sensitive to dataset characteristics and may harm accuracy, the proposed method demonstrates consistent performance across all datasets and metrics, outperforming prior distribution-based models. Nevertheless, challenges remain in aligning confidence estimates with prediction error.

10. **Argus: A Hybrid Edge-Cloud Architecture for Orchestrating Heterogeneous AI in Smart Environments**

    Status: *Submitted*

    Confderence: *23rd IEEE International Conference on Software Architecture (ICSA) 2026*

    Abstract: The paradigm of Smart Environments (SE) has transitioned from simple remote automation to proactive, context-aware ecosystems driven by Artificial Intelligence. However, orchestrating heterogeneous AI subsystems—which range from high-frequency sensor streams to computationally intensive Large Language Models (LLMs)—imposes significant challenges regarding latency, interoperability, and data privacy. In this paper, we propose **Argus**, a distributed, event-driven software architecture designed to orchestrate intelligent decision-making across the Edge-Cloud continuum. We validate the architecture through a reference implementation and performance evaluation, demonstrating that Argus effectively accommodates heavy computational workloads, such as those used in generative AI

and recommendation systems, to create intelligent environments that dynamically adapt to user needs.

11. **Paperman - A Recommender System for Scientific Papers**

    Status: *Developing*

    Journal: *Communications of the Association for Information Systems*

    Abstract: Research Context: In the modern academic landscape, the continuous growth of scientific publications has democratized access to knowledge but also created a challenging environment for researchers. **Scientific and/or Practical Problem**: This flood of information leads to significant information overload, rendering the task of literature review an exhausting and inefficient process. Researchers spend considerable time on manual searches with low-quality returns, as current tools often lack the deep personalization needed to recommend truly relevant articles and fail to integrate seamlessly into the academic workflow. **Proposed Solution and/or Analysis**: Develop a platform to facilitate initial stages of research, through recommendation systems, models based on the researcher's profile, and data post-processing. The proposed system addresses common challenges in academic research, such as information overload and the need for efficient discovery of relevant publications, by leveraging the researcher's profile to provide tailored recommendations. **Related IS Theory**: This paper is related to social network theory and socio-technical theory, by processing publicly available social data to enhance lengthy tasks. **Research Method**: Employment of natural language processing and machine learning techniques to analyze researchers' publication history and generate personalized recommendations for scientific articles. The system architecture includes an API for data collection and processing, integrations with external services, and a browser extension for intuitive presentation of recommendations. **Summary of Results**: Experimental results demonstrate the system's effectiveness through common recommender systems evaluation metrics, such as MRR of 0.8 and nDCG@5 of 0.9407, indicating high relevance of generated recommendations. Another experiment using offline evaluation methods presented better results after implementation improvements, reporting a precision of 0.77, an MRR of 0.86, and an nDCG of 0.92. **Contributions and Impact to the IS area**: The study contributes to the field of educational recommendation systems by offering a practical solution to optimize the literature review process and the discovery of related works in scientific research.

12. **Applying Gaussian Uncertainty-Aware Learning-to-Rank Methods in Recommender Systems**

    Status: *Developing*

    Journal: *IEEE Transactions on Neural Networks and Learning Systems*

    Abstract: Traditional learning-to-rank methods for recommender systems primarily focus on optimizing relevance, often overlooking the model's uncertainty about

its predictions. This lack of awareness of uncertainty limits model interpretability and can lead to unreliable recommendations, particularly in data-sparse or domain-shifted contexts. To address this issue, we propose an uncertainty-aware learning-to-rank framework that integrates confidence estimation directly into the optimization process. The method reformulates the Bayesian personalized ranking loss by replacing the logistic sigmoid with the cumulative distribution function of the Normal distribution. This allows the model to jointly learn both the mean and variance of the predicted relevance scores. The approach is evaluated on four datasets—Amazon Beauty, Jester Joke, MovieLens 1M, and Rotten Tomatoes—using both matrix factorization and deep graph attention network architectures. Experimental results show that the proposed models maintain or improve ranking performance compared to baselines, while achieving well-calibrated confidence estimates that strongly correlate with prediction error. Furthermore, the findings reveal a calibration proportional to the model's overall performance. For instance, a domain-dependent calibration happened, with GAT-based variants excelling in entertainment datasets. Overall, the proposed framework offers a principled approach to incorporating uncertainty into ranking models, thereby enhancing both predictive reliability and interpretability in recommender systems.

9.4 SUMMARY

This chapter summarizes the work's main findings, contributions, and future directions. We benchmarked existing confidence estimation methods in recommender systems, including OrdRec, CPMF, CBPMF, and LBD. The thesis contributions include a comprehensive analysis of prior probabilistic models, the development of confidence-integrated architectures such as PRGAT, and a unified uncertainty-aware ranking formulation. We introduced and evaluated an uncertainty-aware learning-to-rank framework, showing that it achieves strong accuracy, ranking quality, and confidence calibration across diverse datasets. Finally, we outline future research opportunities, including the development of improved optimization strategies, the exploration of broader datasets, the exploration of new confidence estimation approaches, and the integration of uncertainty into sequential and explainable recommendation models.

BIBLIOGRAPHY

ALJUKHADAR, S. S. M.; DAOUST, C.-E. Using recommendation agents to cope with information overload. *International Journal of Electronic Commerce*, Routledge, v. 17, n. 2, p. 41–70, 2012. Disponível em: <https://doi.org/10.2753/JEC1086-4415170202>.

ALVES, R.; LEDENT, A.; KLOFT, M. Uncertainty-adjusted recommendation via matrix factorization with weighted losses. *IEEE Transactions on Neural Networks and Learning Systems*, v. 35, n. 11, p. 15624–15637, 2024.

ANDRADE-RUIZ, G. et al. Emerging perspectives on the application of recommender systems in smart cities. *Electronics*, v. 13, n. 7, 2024. ISSN 2079-9292. Disponível em: <https://doi.org/10.3390/electronics13071249>.

BAHRANI, P. et al. A hybrid semantic recommender system based on an improved clustering. *The Journal of Supercomputing*, Springer, v. 80, n. 9, p. 13341–13385, 2024.

BOBADILLA, J. et al. Recommender systems survey. *Knowledge-Based Systems*, v. 46, p. 109–132, 2013. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705113001044>.

BURGES, C. et al. Learning to rank using gradient descent. In: *Proceedings of the 22nd International Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2005. (ICML '05), p. 89–96. ISBN 1595931805. Disponível em: <https://doi.org/10.1145/1102351.1102363>.

BURKE, R. The adaptive web. In: BRUSILOVSKY, P.; KOBSA, A.; NEJDL, W. (Ed.). Berlin, Heidelberg: Springer-Verlag, 2007. cap. Hybrid Web Recommender Systems, p. 377–408. ISBN 978-3-540-72078-2. Disponível em: <http://dl.acm.org/citation.cfm?id=1768197.1768211>.

CAO, Z. et al. Learning to rank: from pairwise approach to listwise approach. In: *Proceedings of the 24th International Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2007. (ICML '07), p. 129–136. ISBN 9781595937933. Disponível em: <https://doi.org/10.1145/1273496.1273513>.

COSCRATO, V.; BRIDGE, D. Estimating and evaluating the uncertainty of rating predictions and top-n recommendations in recommender systems. *ACM Trans. Recomm. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 1, n. 2, apr 2023. Disponível em: <https://doi.org/10.1145/3584021>.

DENG, D. et al. Neural gaussian mixture model for review-based rating prediction. In: *Proceedings of the 12th ACM Conference on Recommender Systems*. New York, NY, USA: Association for Computing Machinery, 2018. (RecSys '18), p. 113–121. ISBN 9781450359016. Disponível em: <https://doi.org/10.1145/3240323.3240353>.

FENG, C. et al. Attention-based graph convolutional network for recommendation system. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.: s.n.], 2019. p. 7560–7564.

GAO, C. et al. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. Association for Computing Machinery, New York, NY, USA, v. 1, n. 1, mar 2023. Disponível em: <https://doi.org/10.1145/3568022>.

GOHARI, F. S.; ALIEE, F. S.; HAGHIGHI, H. A new confidence-based recommendation approach: Combining trust and certainty. *Information Sciences*, v. 422, p. 21–50, 2018. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025516321545>.

GOLDBERG, K. et al. Eigentaste: A constant time collaborative filtering algorithm. *information retrieval*, Springer, v. 4, n. 2, p. 133–151, 2001.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

HAMILTON, W. L.; YING, R.; LESKOVEC, J. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

HARPER, F. M.; KONSTAN, J. A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 5, n. 4, dez. 2015. ISSN 2160-6455.

HASSAN, T. Trust and trustworthiness in social recommender systems. In: *Companion Proceedings of The 2019 World Wide Web Conference*. New York, NY, USA: Association for Computing Machinery, 2019. (WWW '19), p. 529–532. ISBN 9781450366755. Disponível em: <https://doi.org/10.1145/3308560.3317596>.

HAYKIN, S. *Neural Networks and Learning Machines*. 3. ed. [S.l.]: Prentice Hall, 2008.

HIMABINDU, T. V.; PADMANABHAN, V.; PUJARI, A. K. Conformal matrix factorization based recommender system. *Information Sciences*, v. 467, p. 685–707, 2018. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025516309124>.

HOGG, R. V.; CRAIG, A. T. *Introduction to Mathematical Statistics (Second Edition)*. New York: Macmillan, 1965.

HUANG, L. et al. Broad recommender system: An efficient nonlinear collaborative filtering approach. *IEEE Transactions on Emerging Topics in Computational Intelligence*, v. 8, n. 4, p. 2843–2857, 2024.

KNYAZEV, N.; OOSTERHUIS, H. A lightweight method for modeling confidence in recommendations with learned beta distributions. In: *Proceedings of the 17th ACM Conference on Recommender Systems*. [s.n.], 2023. p. 306–317. Disponível em: <https://doi.org/10.1145/3604915.3608788>.

KOREN, Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2008. (KDD '08), p. 426–434. ISBN 9781605581934. Disponível em: <https://doi.org/10.1145/1401890.1401944>.

KOREN, Y.; BELL, R.; VOLINSKY, C. Matrix factorization techniques for recommender systems. *Computer*, v. 42, n. 8, p. 30–37, 2009.

KOREN, Y.; RENDLE, S.; BELL, R. Advances in collaborative filtering. In: _____. *Recommender Systems Handbook*. New York, NY: Springer US, 2022. p. 91–142. ISBN 978-1-0716-2197-4. Disponível em: <https://doi.org/10.1007/978-1-0716-2197-4_3>.

KOREN, Y.; SILL, J. Ordrec: an ordinal model for predicting personalized item rating distributions. In: *Proceedings of the Fifth ACM Conference on Recommender Systems*. New York, NY, USA: Association for Computing Machinery, 2011. (RecSys '11), p. 117–124. ISBN 9781450306836. Disponível em: <https://doi.org/10.1145/2043932.2043956>.

KWEON, W. *Confidence Calibration for Recommender Systems and Its Applications*. 2024. Disponível em: <https://arxiv.org/abs/2402.16325>.

KWEON, W. *Confidence Calibration for Recommender Systems and Its Applications*. 2024. Disponível em: <https://doi.org/10.48550/arXiv.2402.16325>.

LI, Y. et al. Recent developments in recommender systems: A survey [review article]. *IEEE Computational Intelligence Magazine*, v. 19, n. 2, p. 78–95, 2024. Disponível em: <https://doi.org/0.1109/MCI.2024.3363984>.

LI, Z. et al. Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications. In: *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. [S.l.: s.n.], 2020. p. 1677–1688.

MARQUES, R. P. Sobrecarga de informação na era digital: Causa ou consequência? In: _____. [s.n.], 2016. p. 19–28. ISBN 978-989-95290-9-0. Disponível em: <https://www.researchgate.net/publication/316047521_Sobrecarga_de_Informacao_na_Era_Digital_Causa_ou_Consequencia>.

MAZUROWSKI, M. A. Estimating confidence of individual rating predictions in collaborative filtering recommender systems. *Expert Systems with Applications*, v. 40, n. 10, p. 3847–3857, 2013. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417413000031>.

MESAS, R. M.; BELLOGÍN, A. Exploiting recommendation confidence in decision-aware recommender systems. *Journal of Intelligent Information Systems*, Springer, v. 54, n. 1, p. 45–78, 2020. Disponível em: <https://doi.org/10.1007/s10844-018-0526-3>.

MITCHELL, T. M. *Machine Learning*. New York, NY: McGraw-Hill, 1997. ISBN 978-0070428072.

MOON, J. et al. Confidence-aware learning for deep neural networks. In: III, H. D.; SINGH, A. (Ed.). *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020. (Proceedings of Machine Learning Research, v. 119), p. 7034–7044. Disponível em: <https://proceedings.mlr.press/v119/moon20a.html>.

NETO, F. S. A.; COSTA, A. F. da; MANZATO, M. G. *CoBaR: Confidence-Based Recommender*. 2018. Disponível em: <https://arxiv.org/abs/1808.07089>.

NETO, F. S. A.; COSTA, A. F. da; MANZATO, M. G. *CoBaR: Confidence-Based Recommender*. 2018. Disponível em: <https://doi.org/10.48550/arXiv.1808.07089>.

NIKOLAKOPOULOS, A. N. et al. Trust your neighbors: A comprehensive survey of neighborhood-based methods for recommender systems. In: _____. *Recommender Systems Handbook*. New York, NY: Springer US, 2022. p. 39–89. ISBN 978-1-0716-2197-4. Disponível em: <https://doi.org/10.1007/978-1-0716-2197-4_2>.

PAPADOPOULOS, G.; EDWARDS, P.; MURRAY, A. Confidence estimation methods for neural networks: a practical comparison. *IEEE Transactions on Neural Networks*, v. 12, n. 6, p. 1278–1287, 2001.

PENG, S.; SUGIYAMA, K.; MINE, T. Less is more: Removing redundancy of graph convolutional networks for recommendation. *ACM Trans. Inf. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 42, n. 3, jan 2024. ISSN 1046-8188. Disponível em: <https://doi.org/10.1145/3632751>.

PIYADASA, T. D.; SILVA, R. S. R.; KASTHURIRATHNA, D. Recent developments and limitations in recommender systems: A review. In: *2022 4th Novel Intelligent and Leading Emerging Sciences Conference (NILES)*. [S.l.: s.n.], 2022. p. 81–85.

RENDLE, S. et al. *BPR: Bayesian Personalized Ranking from Implicit Feedback*. 2012. Disponível em: <https://arxiv.org/abs/1205.2618>.

RICCI, F. et al. (Ed.). *Recommender Systems Handbook*. Springer, 2011. ISBN 978-0-387-85819-7. Disponível em: <http://www.springerlink.com/content/978-0-387-85819-7>.

RODGERS, J. L.; NICEWANDER, W. A. Thirteen ways to look at the correlation coefficient. *The American Statistician*, ASA Website, v. 42, n. 1, p. 59–66, 1988. Disponível em: <https://doi.org/10.1080/00031305.1988.10475524>.

SHAIKH, S. et al. Data augmentation and refinement for recommender system: A semi-supervised approach using maximum margin matrix factorization. *Expert Systems with Applications*, v. 238, p. 121967, 2024. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417423024697>.

SMOLA, A.; VISHWANATHAN, S. *Introduction to Machine Learning*. [S.l.]: Cambridge University Press, 2010.

TAO, Z. et al. Mgat: Multimodal graph attention network for recommendation. *Information Processing & Management*, v. 57, n. 5, p. 102277, 2020. ISSN 0306-4573.

VELIčKOVIć, P. et al. *Graph Attention Networks*. 2018. Disponível em: <https://arxiv.org/abs/1710.10903>.

WANG, C. et al. Confidence-aware matrix factorization for recommender systems. In: *Proceedings of the AAAI Conference on artificial intelligence*. [s.n.], 2018. v. 32, n. 1. Disponível em: <https://doi.org/10.1609/aaai.v32i1.11251>.

WANG, H. et al. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. New York, NY, USA: Association for Computing Machinery, 2018. (CIKM '18), p. 417–426. ISBN 9781450360142.

WANG, X. et al. Kgat: Knowledge graph attention network for recommendation. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York, NY, USA: Association for Computing Machinery, 2019. (KDD '19), p. 950–958. ISBN 9781450362016.

WANG, X.; KADIOĞLU, S. Modeling uncertainty to improve personalized recommendations via bayesian deep learning. *International Journal of Data Science and Analytics*, Springer, v. 16, n. 2, p. 191–201, 2023. Disponível em: <https://doi.org/10.1007/s41060-020-00241-1>.

WATT, T. A.; MCCLEERY, R. H.; HART, T. *Introduction to statistics for biology*. [S.l.]: CRC Press, 2007.

WU, S. et al. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 55, n. 5, dec 2022. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3535101>.

XU, E. et al. Limits of predictability in top-n recommendation. *Information Processing Management*, v. 61, n. 4, p. 103731, 2024. ISSN 0306-4573. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0306457324000918>.

YANG, Y.; BUETTNER, F. Multi-output gaussian processes for uncertainty-aware recommender systems. In: CAMPOS, C. de; MAATHUIS, M. H. (Ed.). *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*. PMLR, 2021.

(Proceedings of Machine Learning Research, v. 161), p. 1505–1514. Disponível em: <https://proceedings.mlr.press/v161/yang21a.html>.

YOU, J. et al. Hierarchical temporal convolutional networks for dynamic recommender systems. In: *The World Wide Web Conference*. New York, NY, USA: Association for Computing Machinery, 2019. (WWW '19), p. 2236–2246. ISBN 9781450366748. Disponível em: <https://doi.org/10.1145/3308558.3313747>.

ZHANG, M.; GUO, X.; CHEN, G. Prediction uncertainty in collaborative filtering: Enhancing personalized online product ranking. *Decision Support Systems*, v. 83, p. 10–21, 2016. ISSN 0167-9236. Disponível em: <https://doi.org/10.1016/j.dss.2015.12.004>.

ZHANG, S. et al. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 52, n. 1, feb 2019. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3285029>.

ZHAROVA, A. et al. An explainable multi-agent recommendation system for energy-efficient decision support in smart homes. *Environmental Data Science*, v. 3, p. e7, 2024. Disponível em: <https://doi.org/10.1017/eds.2024.8>.