UNIVERSITE CHADLI BENDJEDID
FACULTY OF SCIENCE AND TECHNOLOGY
COMPUTER SCIENCE DEPARTMENT

Thesis Of The End Of Studies project

In preparation for obtaining the diploma

MASTER

Intelligent Computer Systems

Under the subject :

Addressing Sparsity Problem In E-Commerce Recommender Systems

---

Presented by :
Mr. Abbas Djaber

On June 22, 2024, before the jury composed of :

- President	- Mme. MAATALLAH Madjda	MCB	Chadli BenDjedid
- Examiner	- Mme. ZIANI Amel	MCB	Chadli BenDjedid
- Supervisor	- Mme. GASMI Ibtissem	MCA	Chadli BenDjedid
- Co-Supervisor	- Mr. KHANTOUCHI Ramzi	/	Chadli BenDjedid

Academic year 2023-2024

Abstract

Recommendation systems are essential in e-Commerce, greatly improving the user experience through personalised product suggestions and increasing sales. Despite their importance, these systems often face the challenge of data sparsity, where users engage only with a limited subset of the available product catalogue. This parsimony prevents the system from accurately predicting user preferences, which reduces the quality of recommendations.This study focuses on the problem of the scarcity of ecommerce recommendation systems, examining its effects on recommendation accuracy and exploring various strategies to overcome this challenge. Specifically, it uses the latent Dirichlet allocation (LDA) for subject modeling to improve the representation of rare data and exploits convolutional neural networks (CNN) to improve the recommendation process. By addressing the issue of scarcity, this study aims to improve the performance of recommendation systems, thereby increasing user satisfaction, engagement, and ultimately revenue growth for e-Commerce platforms.We evaluate the proposed approach using measures such as Mean Absolute Error (EAM). The results highlight the importance of addressing data scarcity to improve the precision and relevance of recommendations, contributing to a more personalised and engaging user experience in e-commerce environments.

Résumé

Les systèmes de recommandation sont essentiels dans le commerce électronique, améliorant considérablement l'expérience utilisateur grâce à des suggestions de produits personnalisées et stimulant les ventes. Malgré leur importance, ces systèmes sont souvent confrontés au défi de la rareté des données, où les utilisateurs ne s'engagent qu'avec un sous-ensemble limité du catalogue de produits disponibles. Cette parcimonie empêche le système de prédire avec précision les préférences des utilisateurs, ce qui diminue la qualité des recommandations. Cette étude se concentre sur le problème de la rareté des systèmes de recommandation du commerce électronique, en examinant ses effets sur la précision des recommandations et en explorant diverses stratégies pour surmonter ce défi. Plus précisément, elle utilise l'allocation Dirichlet latente (LDA) pour la modélisation des sujets afin d'améliorer la représentation des données rares et exploite les réseaux de neurones convolutifs (CNN) pour améliorer le processus de recommandation. En abordant la question de la rareté, Cette étude vise à améliorer la performance des systèmes de recommandation, augmentant ainsi la satisfaction des utilisateurs, l'engagement et, finalement, la croissance des revenus pour les plateformes de commerce électronique. Nous évaluons ainsi l'approche proposée à l'aide de mesures telles que l'erreur absolue moyenne (EAM). Les résultats soulignent l'importance de s'attaquer à la rareté des données pour améliorer la précision et la pertinence des recommandations, contribuant ainsi à une expérience utilisateur plus personnalisée et plus engageante dans les environnements de commerce électronique.

خلاصة

تعد أنظمة التوصية ضرورية في التجارة الإلكترونية، مما يؤدي إلى تحسين تجربة المستخدم بشكل كبير من خلال اقتراحات المنتجات الشخصية وزيادة المبيعات. وعلى الرغم من أهميتها، غالبا ما تواجه هذه الأنظمة تحدي ندرة البيانات، حيث يتفاعل المستخدمون فقط مع مجموعة فرعية محدودة من كتالوج المنتجات المتاح. يمنع هذا التحليل النظام من التنبؤ بتفضيلات المستخدم بدقة، مما يقلل من جودة التوصيات. تركز هذه الدراسة على مشكلة ندرة أنظمة توصيات التجارة الإلكترونية، ودراسة آثارها على دقة التوصيات واستكشاف استراتيجيات مختلفة للتغلب على هذا التحدي. على وجه التحديد، يستخدم تخصيص ديريتشليت الكامن لنمذجة الموضوع لتحسين تمثيل البيانات النادرة واستغلال الشبكات العصبية التلافيفية لتحسين عملية التوصية. من خلال معالجة مسألة الندرة، تهدف هذه الدراسة إلى تحسين أداء أنظمة التوصية، وبالتالي زيادة رضا المستخدمين والمشاركة، وفي نهاية المطاف، نمو إيرادات منصات التجارة الإلكترونية. نقوم بتقييم النهج المقترح باستخدام تدابير مثل متوسط الخطأ المطلق . وتسلط النتائج الضوء على أهمية معالجة ندرة البيانات لتحسين دقة التوصيات وأهميتها، مما يسهم في جعل تجربة المستخدم أكثر تخصيصا وجاذبية في بيئات التجارة الإلكترونية.

Acknowledgments

I would like to begin by thanking Allah, for His blessings and guidance, and providing me with the men and women who have supported me along the way. I am deeply grateful to my family, starting with my father, who raised me with the strength and wisdom to become the man I am today. His being my role model is unwavering since I became aware of my existence. My heartfelt thanks go to my mother for her tireless concern and dedication to my well-being, especially during the challenging times of my work on this thesis and even before. I owe a significant debt of gratitude to my brother, Anis, who has been an immense support in every sense of the word. Additionally, I wish to thank my beloved sisters for their companionship and encouragement. A special thanks to my aunts and uncles for their love and support throughout my life. I also wish to acknowledge the invaluable guidance provided by my supervisor, Mme. Gasmi Ibtissam, and the co-supervisor, Ramzi Khantouchi. Their expertise and support were instrumental in helping me navigate the complexities of this thesis.
I am deeply indebted to all those mentioned above for their generosity, expertise, and support.

Contents

List of Figures

List of Tables

General Introduction

Recommender systems have become integral to modern digital experiences, facilitating personalized recommendations across various platforms from e-commerce to social media and beyond. These systems alleviate the overwhelming volume of choices users face by predicting their preferences and suggesting items of potential interest. This introduction explores the foundational concepts, evolution, types, challenges, and applications of recommender systems. Initially conceived to address information overload, recommender systems have evolved significantly, driven by advances in machine learning, data analytics, and computational power. They primarily operate through three main paradigms: collaborative filtering, content-based filtering, and hybrid approaches that combine both. Context-aware systems further enhance recommendation accuracy by incorporating additional user and item context such as location or time.

Despite their benefits, recommender systems encounter several challenges. These include data sparsity, where not all users have sufficient interaction data; the cold start problem, which affects new users or items lacking sufficient data for accurate recommendations; scalability issues with large datasets; and concerns about privacy and security. To mitigate these challenges, various techniques have been developed, ranging from sophisticated algorithms to novel evaluation metrics such as Mean Absolute Error (MAE) and Precision-Recall. These metrics gauge the effectiveness of recommendations and help refine the performance of the system.

The applications of recommender systems span various sectors, including e-Commerce for product recommendations, streaming services for personalized content delivery, social networks for friend or content suggestions, and education for adaptive learning paths. Each sector leverages recommender systems to improve user engagement, satisfaction, and overall experience.

This thesis sets the stage for exploring the intricate workings of recommender systems, highlighting their pivotal role in shaping user interactions and content consumption in today's digital landscape. By understanding their mechanisms, challenges, and applications, we can appreciate both the opportunities and complexities involved in designing and deploying effective recommender systems.

The focus of this work is on developing a recommender system that addresses the sparsity problem in the field of e-commerce. The sparsity problem arises when there is insufficient interaction data available for users or items, making it challenging to generate accurate recommendations. To tackle this issue, this thesis proposes a hybrid approach that combines Latent Dirichlet Allocation (LDA) topic modeling and deep learning techniques using Convolutional Neural Networks (CNNs). LDA helps in understanding the latent topics or themes within the items, while CNNs leverage these topics to enhance the recommendation process by capturing intricate patterns and features from the data. The thesis is structured into three chapters. The first chapter delves into the foundations and advances in recommender systems, providing a comprehensive literature review that covers the evolution of these systems, their various types, and the challenges they face. This chapter sets the groundwork for understanding the current state of recommender systems and the innovations driving their development. The second chapter focuses on the motivation and concept behind the proposed work. It explains the rationale for choosing a hybrid approach that combines LDA and CNNs, detailing how this combination can effectively address the sparsity problem. This chapter also outlines the methodological framework and the specific techniques employed in developing the recommender system. The third chapter presents the results of the study, showcasing the performance and effectiveness of the proposed system. It includes a thorough evaluation of the system using various metrics, demonstrating how the hybrid approach improves recommendation accuracy and user satisfaction compared to traditional methods. Finally, the thesis concludes with a summary of the findings and discusses potential directions for future research. This conclusion reflects on the contributions of the work to the field of recommender systems and highlights areas where further advancements can be made to enhance the capabilities and applications of these systems in e-commerce and beyond.

Chapter 1

Foundations and Advances in Recommender Systems

1.1 Introduction

Recommender systems are essential in e-Commerce, offering personalised product suggestions to enhance user experience and drive sales. However, these systems often face the challenge of data sparsity, where users interact with only a small subset of the available products. This limits the system's ability to generate accurate recommendations, affecting user satisfaction and engagement.

This chapter addresses the sparsity problem in e-commerce recommendation systems, examining its impact on recommendation quality, and exploring strategies to mitigate it. Improving recommendation accuracy despite sparse data is crucial for increasing user satisfaction, engagement, and ultimately revenue growth for e-commerce businesses.

The chapter begins by highlighting the importance of recommender systems in e-commerce and the challenges posed by data sparsity. It then outlines research objectives focused on understanding and addressing these challenges, aiming to enhance the performance and effectiveness of recommendation systems.

1.2 Introduction to Recommender Systems

In order to promote a better understanding of the functionality and effectiveness of recommender systems, this study aims to explore the complexities of these systems by clarifying their underlying processes and operational dynamics.

1.2.1 Concept and Importance

recommender systems, are sophisticated algorithms designed to analyze user preferences and provide personalized suggestions for items or content [41]. These systems have become ubiquitous on modern digital platforms, ranging from e-Commerce sites and streaming services to social media platforms and news websites. Using data on user behaviour, item attributes, interactions, and then presenting the goods that consumers are interested in, recommendation systems have simplified the customer experience, helping to convert them from potential customers into real customers [105]. The significance of RS also lies in their ability to influence human views and consumer behaviour, creating biases that can impact user choices and even cause inequality.[30]

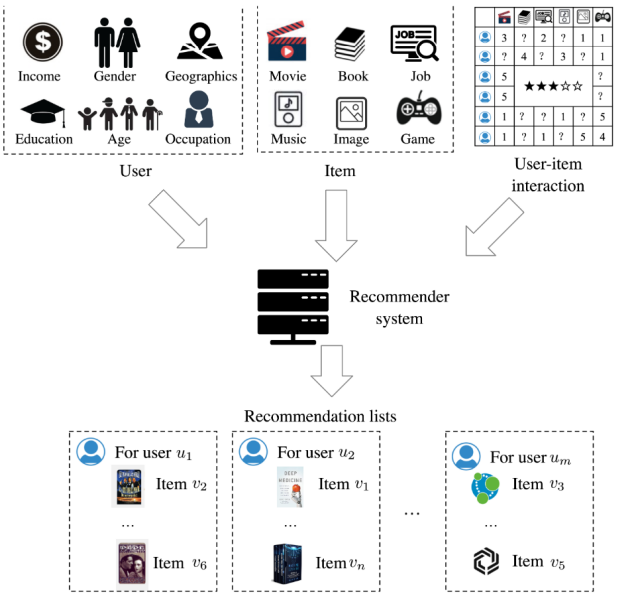

Figure 1.1: A graphical illustration of a recommender system [6].

1.2.2 Evolution of Recommender Systems

As mentioned in [38] the evolution of recommender systems has been marked by significant milestones and advancements.

Classic Recommender systems

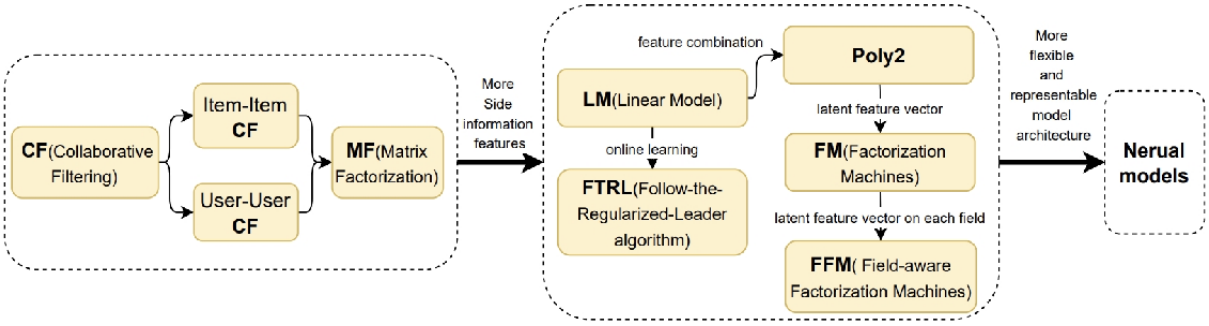

Figure 1.2: evolution graph of Classic RS[38].

In the early foundations of recommender systems during the 1990s, future developments in the field of information retrieval and filtering were made possible by the groundbreaking work of Belkin and Croft [20]. Approximately the same time, recommender technology underwent a significant step forward with the introduction of the Tapestry system by Goldberg et al., one of the first examples of collaborative filtering-based information filtering systems [44]. Furthermore, GroupLens, a news recommendation service built on user-user collaborative filtering, was created by researchers from MIT and UMN to demonstrate the potential of collaborative filtering techniques in personalized content recommendations [90]. This period also witnessed the emergence of other recommendation systems like Ringo for music and Video Recommender for video recommendations [102, 54]. As the 1990s progressed into the mid to late years, the commercialisation of recommender systems gained momentum, with companies such as Net Perceptions recognising the business opportunities they presented. Schafer et al. further emphasised the value of recommender systems in increasing sales for e-Commerce platforms, underlining their practical significance [99]. Moving into the early 2000s, the convergence of academic and industrial efforts became evident with the launch of the MovieLens project by the GroupLens research lab, which provided a valuable dataset for recommendation studies [49]. Collaborative filtering technologies, including item-item and user-user collaborative filtering, as well as Singular Value Decomposition (SVD) based approaches, dominated the landscape of recommender systems during this period [22, 53, 72, 97, 98]. The Netflix Prize, which ran from 2006 to 2009, spurred significant research into matrix factorization models, which became pivotal for rating prediction tasks [66, 67]. Toward the late 2000s, there was a notable shift towards user-centric evaluation metrics, culminating in the establishment of the first ACM Recommender Systems Conference in 2007 [9]. Richardson and al.'s introduction of a logistic regression (LR) model further underscored the importance of user-centric approaches, leading to notable advancements in click-through rate estimation [91].

Deep Learning Recommender systems

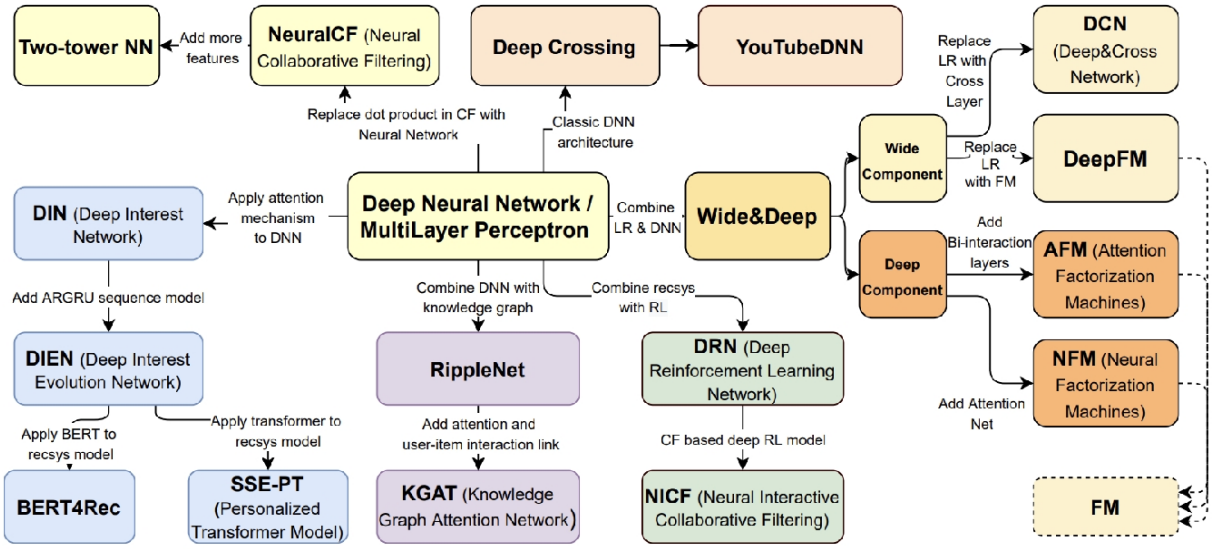

Figure 1.3: evolution graph of Deep learnig RS system [38].

In the deep learning era of the 2010s, recommender systems experienced a paradigm shift with the emergence of deep learning-based models. Rendle's proposal of Factorization Machines (FMs) in 2010 marked a significant milestone, as FMs combined the strengths of Support Vector Machines (SVM) with factorization models, offering improved performance and flexibility [89]. This paved the way for the adoption of deep learning techniques in recommender systems, both in academia and industry.

Deep learning-based recommendation models began to proliferate, offering advanced capabilities to capture complex patterns and relationships in data. Models such as Wide&Deep, DeepFM, and YouTubeDNN gained prominence for their ability to handle large-scale datasets and deliver accurate recommendations [27, 47, 33]. These models leverage deep neural networks to learn intricate feature representations, enabling them to capture latent features and user preferences more effectively.

Furthermore, researchers proposed a variety of deep recommendation models tailored to different use cases and data characteristics. Models like FNN (Factorization-based Neural Networks), PNN (Product-based Neural Networks), NeuralCF (Neural Collaborative Filtering), NFM (Neural Factorization Machines), and CVAE (Collaborative Variational Autoencoder) offered innovative approaches to address various challenges in recommendation tasks [125, 88, 52, 51, 71]. These models leverage techniques such as neural networks, attention mechanisms, and variational autoencoders to improve the precision and personalisation of recommendations.

1.3 Types of Recommender Systems

Recommender systems can be classified into several types based on their underlying algorithms, data sources, and the nature of recommendations they provide. Here are some common types:

1.3.1 Collaborative Filtering Systems

One of the main techniques used by recommendation engines is collaborative filtering, which powers the predictive algorithms that generate the recommendations that these systems make. Through the examination and manipulation of consumer information, it attempts to suggest products that correspond to their tastes. These algorithms evaluate reviews and previous purchases made by a consumer to find like-minded customers. They then suggest products that those similar users have found to be appealing [58].

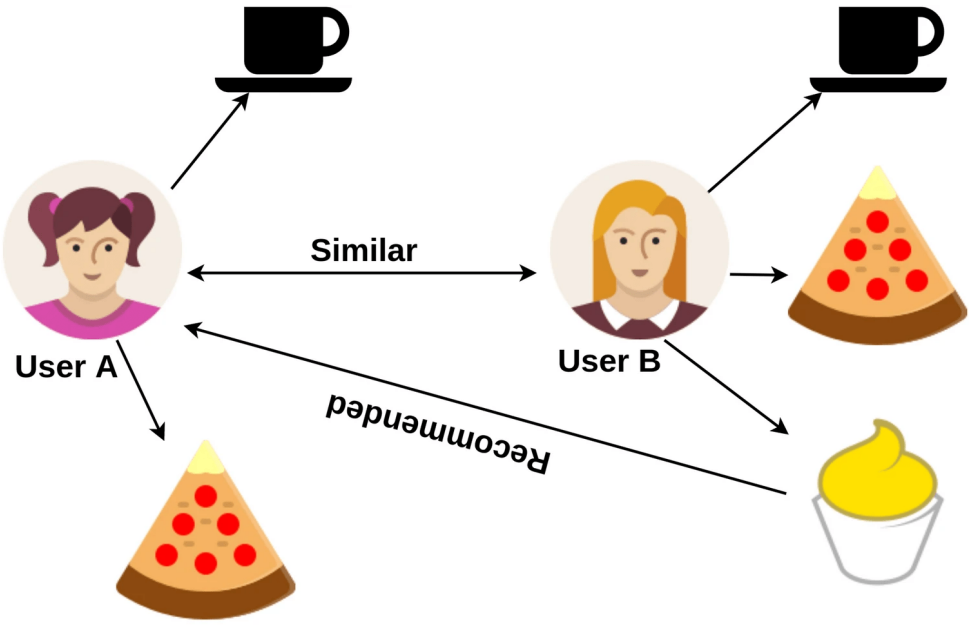

Figure 1.4: A representation of collaborative filtering [19].

Memory-Based Collaborative Filtering

Memory-based recommendation, additionally referred to as neighborhood-based CF, falls into two categories: user-based CF and item-based CF. The basic concept of user-based approaches is that consumers with similar choices in the past are likely to have similar interests in the future. On the other hand, item-based approaches function under the premise that a user might potentially value goods that are comparable to those they have already found appealing. Therefore, the main goal of memory-based CF algorithms is to solve the similarity problem by figuring out how similar two things or people are based on their previous encounters. The basis of memory-based CF is this similarity computation, which allows the system to find relevant neighbours (things or users) and make personalised suggestions based on that identification [42].

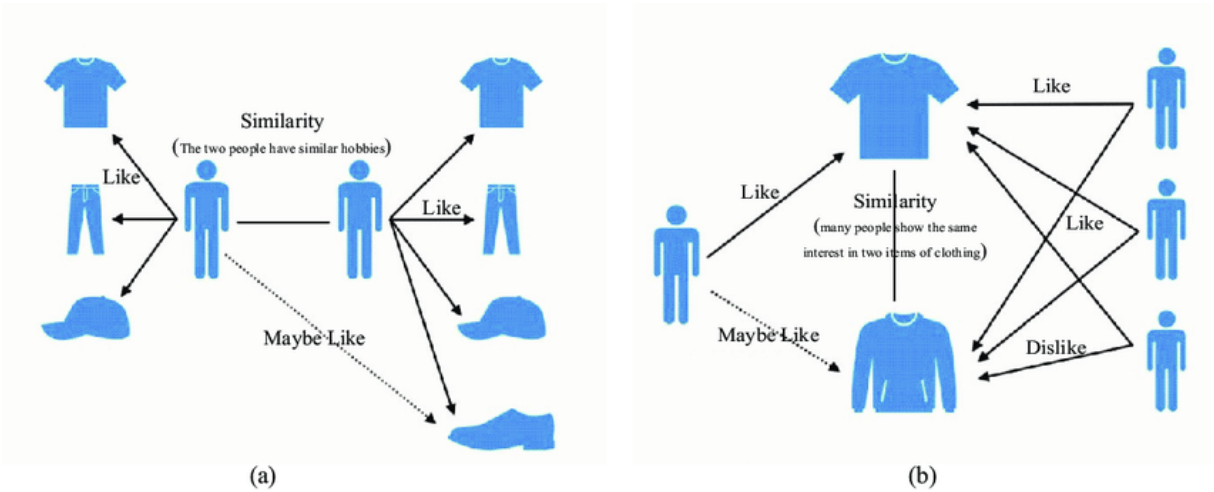

Figure 1.5: user-based & item-based CF [83].

When it comes to methods for measuring similarity between entities, there are many different approaches available:

Table 1.1: Symbols used in measurements of similarity [61]

Notation	Description
$r_{u_a,i}$	The rating given by user u_a on item i
$r_{u_b,i}$	The rating given by the user u_b on the item i
r_{u_a}	The average rating value of the user u_a
r_{u_b}	The average rating value of the user u_b
n	Total number of items
i'	Total number of co-rated items
r_{med}	Median value in rating scale
$\|I_{u_a}\|$	The cardinality of items rated by the user u_a
$\|I_{u_b}\|$	The cardinality of items rated by user u_b

Pearson Correlation Coefficient:
is, in fact, among the most often used conventional similarity metrics in cooperative filtering [36]. By comparing user or item ratings, it evaluates the linear correlation between them, taking into account only co-rated objects. The ratio of two users' (or items') covariance to the product of their standard deviations is the mathematical representation of PCC [61].

$$Sim(u_a, u_b) = \frac{\sum_{i=1}^{i'}(r_{u_a,i} - r_{\bar{u_a}})(r_{u_b,i} - r_{\bar{u_b}})}{\sqrt{\sum_{i=1}^{i'}(r_{u_a,i} - r_{u_a})^2} \cdot \sqrt{\sum_{i=1}^{i'}(r_{\bar{u_b},i} - r_{\bar{u_b}})^2}} \tag{1.1}$$

Cosine Correlation Coefficient:
The cosine angles created between the user rating vectors are evaluated by the cosine similarity measure [119], which measures similarity. Smaller angles are correlated with more similarity and vice versa. There is no need to center the data or modify preference values when using this metric, which is called uncentered cosine similarity.[61]

$$\text{Sim}(u_a, u_b) = \frac{\sum_{i=1}^{i'} r_{u_{a,i}} \cdot r_{u_{b,i}}}{\sqrt{\sum_{i=1}^{n}(r_{u_{a,i}})} \cdot \sqrt{\sum_{i=1}^{n}(r_{u_{b,i}})}} \tag{1.2}$$

Jaccard Similarity Coefficient:
The Jaccard similarity coefficient is commonly called the Tanimoto coefficient similarity measure. The number of common ratings that two users have is the only factor taken into account when calculating similarity. The best results from this strategy come from a large number of common ratings [61].

$$Sim(u_a, u_b) = \frac{|I_{u_a}| \cap |I_{u_b}|}{|I_{u_a}| \cup |I_{u_b}|} \tag{1.3}$$

Euclidean Distance:
This method basically relies on distance, with two processes involved in estimating similarity: first, using an equation to determine the users' distance from one another, and then using the same equation to calculate the similarity. This measure does not produce negative similarity values, much as cosine similarity. Rather, the similarity between users grows as the distance value lowers [61]. Here is how the distance is computed:

$$Dis(u_a, u_b) = \sum_{i=1}^{i'} |r_{u_{a,i}} - r_{u_{b,i}}|^2 \tag{1.4}$$

Model-Based Collaborative Filtering

Model-Based Collaborative Filtering encompasses a variety of machine learning techniques. This includes matrix factorization methods and deep learning approaches.
Matrix Factorisation Techniques:
The factorisation matrix [70, 95] can be easily described as one group of m entities corresponds to another group of n entities. For example, recommending movies to users is common. In this scenario, the goal is to assess the similarity between a user and a movie as if they were of the same category. To do this, both movies and users are represented as vectors of the same dimension, allowing comparison and

matching based on their features [122]. Here are some commonly used matrix factorization techniques in model-based collaborative filtering:

- Singular Value Decomposition (SVD) which is a popular approach for decomposing a matrix into numerous component matrices, revealing several useful and intriguing aspects of the original matrix. We can use SVD to identify the rank of the matrix, measure the susceptibility of a system that is linear to numerical error, and create an ideal approximation of the lower rank to the matrix [28].
- Nonnegative matrix factorization (NNMF), a formal mathematical technique for dimensionality reduction. NNMF represents the original high-dimensional feature vectors as the N-dimensional rows of the input matrix X of n by N, where n is the number of data instances in the training set. Dimensionality is reduced by dividing the matrix X into two matrices, $X = A \cdot B$, where A is n-by-K and B is K-by-N. In this factorization, K represents the number of new features in each of the n feature vectors. Because K is usually fewer than N, each data point is now defined in a lower-dimensional feature space. The new K-dimensional feature vectors are known as latent representations of the original N-dimensional feature vectors [87].

Deep Learning Approaches

- **Deep Matrix Factorization (Deep MF)**, influenced by advancements of deep learning, enables for the collection of hierarchical features, providing useful insights across a wide range of applications. Instead than depending on a single matrix W for approximation, Deep MF uses a sequence of matrices Wl (where l = 1,..., L). Constraining the elements of this decomposition is critical to preserving the integrity and interpretability of the model [48].

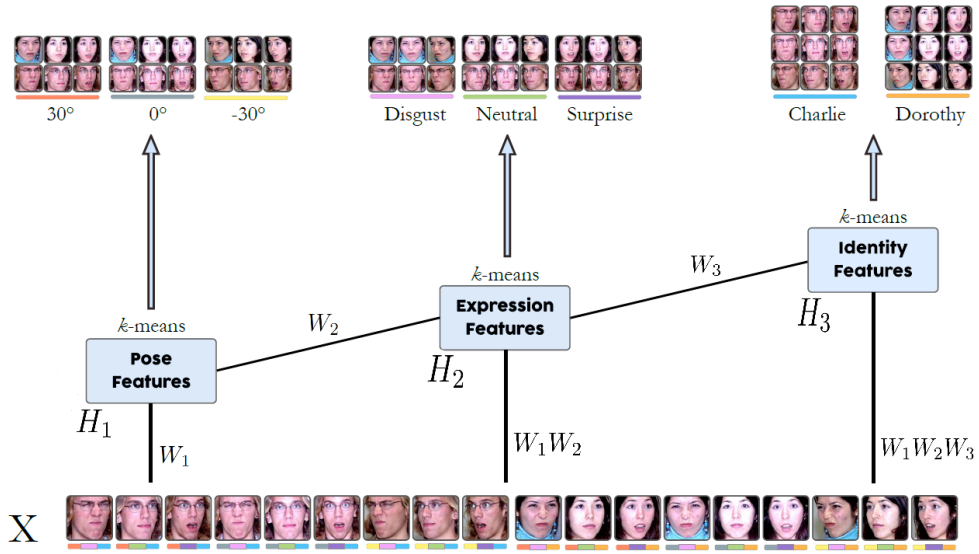

Figure 1.6: Hierarchical features extracted by deep MF on the CMU-PIE face dataset. At each layer $l(l = 1, 2, 3)$, the columns of the representation matrix U^l are clustered according to k-means [113].

- **Autoencoders**, a complex neural network composed of two symmetrical deep-belief networks, each with four or five shallow layers. One side encodes data, the other decodes it. Autoencoders, as opposed to principal component analysis (PCA), provide more versatility by encoding with both linear and nonlinear transformations. They learn important data characteristics by decreasing the difference between the input and output data. The output layer has the same number of neurones as the input layer. There are several different varieties of autoencoders, each with its own set of features such as Sparse Autoencoders, Contractive Autoencoders (CAE), and Complete Autoencoders [13].

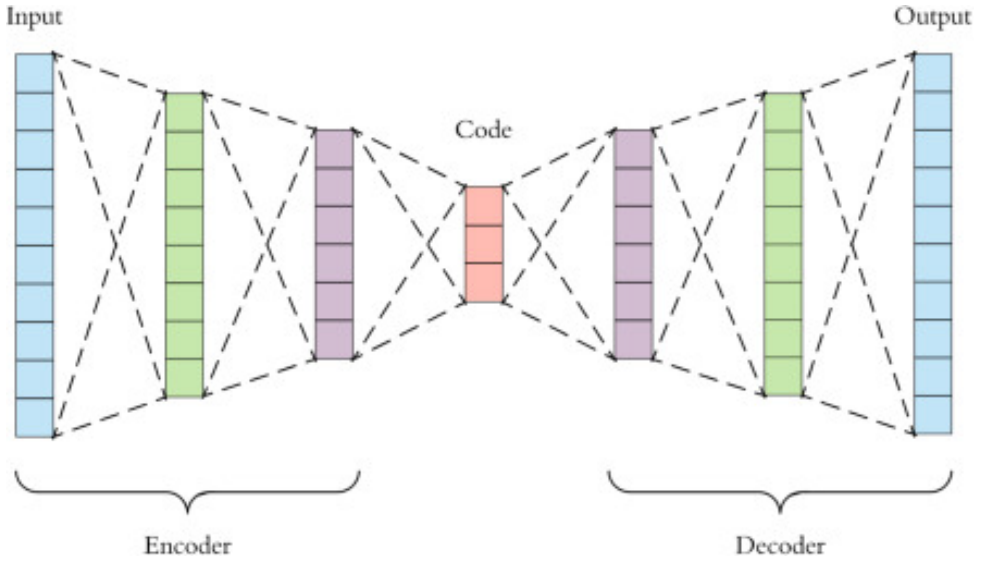

Figure 1.7: Illustration of an Autoencoder [13].

- **Convolutional Neural Networks (CNN)**, Recommender systems frequently confront sparse feedback matrices, which reduces suggestion quality. In response, supplementary data such as product photos, reviews, and audio are critical for improving accuracy. Figure 6 categorises CNN-based recommender systems according to the type of input they use, ranging from pictures to text and audio. This supplementary information, which excludes explicit feedback, improves recommendation algorithms. CNN's capacity to extract audio elements, for example, helps in understanding user profiles, particularly in the music sector. Furthermore, implicit feedback may be used in conjunction with explicit feedback to create more effective recommender systems, resulting in hybrid techniques that combine different data sources to provide better suggestions [85].

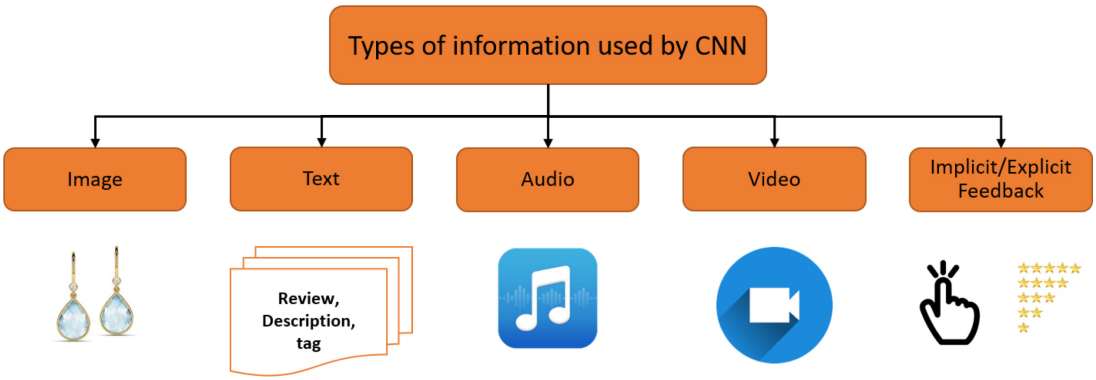

Figure 1.8: Inputs that CNN take [85].

1.3.2 Content-Based Filtering Systems

Definition and Basic Concepts

Content-based recommendation algorithms are a very simple class of algorithms that essentially provide suggestions by comparing the attributes of the items to the preferences of the consumers. In the beginning, it was mostly based on specialised content, such as text, photographs, or music. Their strategy consists of offering things to consumers that display content similar to previously chosen items, as indicated by the past preferences of users [35, 80]. The fundamental stages for content-based recommendation are the following:

1. Create an object feature representation based on its content.
2. Create a user-feature representation based on their traits and actions.

3. Produce a suggestion list based on the amount of alignment between the target item's features and those of the user [115].

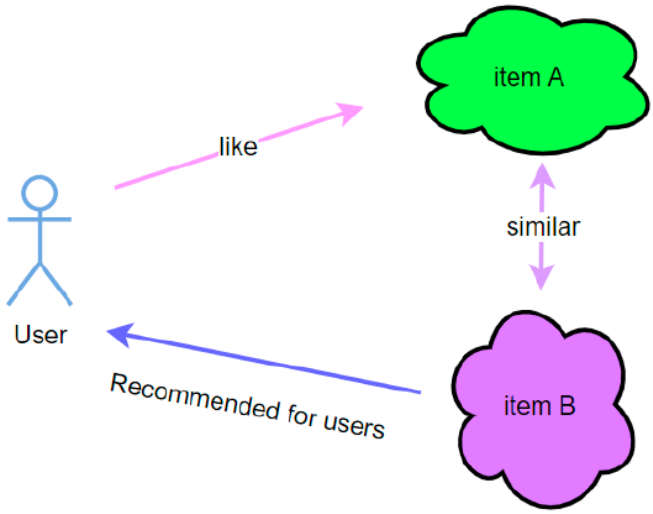

Figure 1.9: Illustratio of content-based recommendation [127].

Content-Based Filtering Techniques

1. **TF-IDF (term frequency-inverse document frequency)**, it displays textual texts as vectors, which helps with classification, grouping, and retrieval tasks. Each document d_i is represented as an n-dimensional vector $(\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in})$ based on the words in the corpus $T = \{t_1, \ldots, t_n\}$. The weight x_{ij} of term t_j in document d_i is the product of its term frequency TF_{ij} and the inverse document frequency IDF_j. The IDF is calculated as $\log\left(\frac{N}{\mathrm{DF}_j}\right)$, where N is the total number of documents and DF_j is the number of documents that include the word t_j. This formula creates TF-IDF vectors for documents [96].

2. **Vector Space Model:** Documents and queries are represented using the VSM in a high-dimensional vector space, where each dimension is mapped to a vocabulary term:

   Document Representation: Documents are represented as n-dimensional vectors. For example, the vector representation of Document D_1 denoted as D_1 is:

   $$D_1 = (w_{11}, w_{12}, w_{13}) \tag{1.5}$$

   Query Representation: Queries are also represented as n-dimensional vectors. For instance, the vector representation of a query Q denoted as Q is:

   $$Q = (w_1, w_2, w_3) \tag{1.6}$$

   Unit Vector Transformation: The unit vector d_1 for Document D_1 is obtained by normalizing its vector representation:

   $$d_1 = \frac{D_1}{\|D_1\|} \tag{1.7}$$

   Similarly, the unit vector q for the query Q is obtained by normalizing its vector representation:

   $$q = \frac{Q}{\|Q\|} \tag{1.8}$$

   Similarity Computation: The similarity between Document D_2 and Query Q is computed using the dot product between their corresponding unit vectors.

   $$\mathrm{sim}(D_2, Q) = d_2 \cdot q \tag{1.9}$$

The dot product is the sum of the product of the corresponding vector components:

$$\text{sim}(D_2, Q) = w_{21} \cdot w_1 + w_{22} \cdot w_2 + w_{23} \cdot w_3 \tag{1.10}$$

This similarity measure serves to determine the order of documents in the collection with regard to a query since it is equivalent to the cosine of the angle between d_2 and q [46].

3. **Word embeddings:** a popular machine learning approach that treats individual words as vectors. These vectors are built in such a manner that their geometric connections convey semantic similarities as well as connections within the words they represent [43].

Latent Dirichlet Allocation (LDA) for Content-Based Filtering

Table 1.2: Explanation of Symbols and Variables

Symbol/Variable	Description
d	Document
θ_d	Mixture of topics in document d
k	Topic index
ϕ_k	Distribution over words for topic k
α	Hyperparameter controlling sparsity of topic distributions in documents.
β	Hyperparameter controlling sparsity of word distributions in topics
w	Word
$z_{d,n}$	Topic assignment for word w in document d
$p(\theta_d\|\alpha)$	Probability distribution of topic proportions in document d
$p(\phi_k\|\beta)$	Probability distribution of word distributions for topic k
$p(w\|\text{user})$	Probability of word w given user preferences

Latent Dirichlet Allocation (LDA) is a generative statistical model used for topic modeling, particularly in the context of content-based filtering in recommendation systems. It operates on the assumption that each document d is represented as a mixture of topics, denoted by θ_d, and each topic k is characterised by a distribution over words, denoted by ϕ_k. Mathematically, this can be expressed as:

$$p(\theta_d|\alpha) = \text{Dirichlet}(\theta_d|\alpha) \tag{1.11}$$

$$p(\phi_k|\beta) = \text{Dirichlet}(\phi_k|\beta) \tag{1.12}$$

where α and β are hyperparameters that control the sparsity of topic distributions in documents and word distributions in topics, respectively.

The generative process of LDA involves:

1. For each document d:
   - Sample topic proportions θ_d from a Dirichlet distribution with parameter α.
2. For each word w in document d:
   - Sample a topic assignment $z_{d,n}$ from the multinomial distribution θ_d.
   - Sample a word w from the multinomial distribution $\phi_{z_{d,n}}$.

The goal of LDA is to infer the posterior distributions $p(\theta_d, \phi_k|\text{corpus})$ given the observed documents. This is typically done using variational inference or Gibbs sampling.

In the context of recommendation systems, LDA can be applied to analyse the textual content associated with the items to extract latent topics. These topics represent underlying themes or concepts present in the textual data. Once the topics are inferred using LDA, recommendations can be made by recommending items that share similar topic distributions. This can be formalized as:

$$p(w|\text{user}) = \sum_{k=1}^{K} p(w|\phi_k)p(\phi_k|\text{user}) \tag{1.13}$$

where $p(w|\phi_k)$ is the probability of the word w given topic k, and $p(\phi_k|\text{user})$ is the probability of topic k given the user's preferences [63].

1.3.3 Hybrid Recommender Systems

Concept and Definition

According to [14], a hybrid recommendation system combines collaborative filtering recommendation systems with content-based recommendation systems to obtain exact performance while mitigating the shortcomings of traditional recommendation methods. The hybrid recommendation system uses the rating matrix in conjunction with user-provided side information, comments, ratings, connections, qualities, and reviews to improve accuracy and performance .

Classification of Hybrid Systems

According to Robin Burke [23] Because hybrid systems avoid the limitations of standalone systems, they perform better. He also classified them into seven types:

1. **Weighted Hybrid Systems:** By giving weights to the output of each recommendation approach, a weighted hybrid recommender system integrates numerous strategies for making recommendations. These weights establish how much of an impact each method has on the overall recommendation score for a particular item. For instance, the P-Tango system [75] starts off giving collaborative and content-based recommenders the same weight, but then modifies these weights in response to how well user rating forecasts turn out. Another method, called Pazzani's combination hybrid [86], combines the recommendations of each recommender into a collection of votes using a consensus procedure. A weighted hybrid system has the advantage of utilizing the best aspects of many recommendation methodologies, resulting in a recommendation process that is more thorough and flexible.

2. **Switching Hybrid System:** dynamically selects between different recommendation techniques based on certain criteria. For example, the DailyLearner system first uses a content-based recommendation method. If this method cannot confidently make a recommendation, then a collaborative recommendation is attempted. This approach addresses the "new user" problem by using a nearest-neighbor content-based technique that doesn't require a large number of examples for accurate classification. Collaborative filtering then provides the ability to recommend items across genres, even if they are not semantically similar to previously rated items.
   Tran and Cohen [112] proposed a similar switching hybrid where the agreement between a user's past ratings and the recommendations of each technique determines which technique to use for the next recommendation. Although switching hybrids introduce complexity by requiring criteria for switching between techniques, they offer the advantage of being sensitive to the strengths and weaknesses of each recommender. This allows the system to adapt its recommendation strategy based on the performance of individual techniques and user preferences.

3. **Mixed Hybrid System:** A 'mixed' hybrid recommender system presents recommendations from multiple techniques together, which is particularly practical when making numerous recommendations simultaneously. For instance, the PTV system [106] combines content-based and collaborative filtering recommendations to assemble television viewing schedules. This approach tackles the 'new item' start-up issue by leveraging content-based methods to recommend new shows based on descriptions. However, it doesn't entirely solve the 'new user' start-up problem. The mixed hybrid can unearth niche items overlooked by a strict content-based approach and often prioritizes content-based recommendations over collaborative ones.

4. **Feature Combination Hybrid System:** The output of one recommender is treated as supplementary feature data in this kind of hybrid recommender system, and it is integrated into the workflow of the other recommender. The recommender that heavily relies on item characteristics (often content-based) usually uses the other recommender's output as supplementary data.
   When using a mix of content-based filtering and collaborative filtering, the system is not entirely

dependent on the collaborative data generated by CBF. Rather, the content-based recommender uses this output as additional data to create the final recommendation list. This method reduces susceptibility to potential problems with data sparsity in the original data collection. It basically makes use of the advantages of both approaches to improve suggestion performance [129].

5. **Cascade Hybrid System:** The cascade hybrid recommender system involves a staged process where recommendations are refined through successive steps. Initially, a recommendation technique is used to generate a preliminary ranking of candidates. Then, a second technique is employed to further refine the recommendations from this candidate set.
   This approach differs from previous hybridisation methods in which multiple techniques are simultaneously integrated or where one technique's output serves as additional data for another. Instead, in the cascade hybrid, each technique is applied sequentially, with the output of one technique informing the subsequent refinement process carried out by another technique.
   Using a cascade approach, the system can capitalise on the strengths of each technique at different stages of the recommendation process. This staged refinement allows for more targeted and personalised recommendations, ultimately improving overall recommendation quality.

6. **Feature Augmentation Hybrid System:** The methodology involves utilising the output of one recommendation technique to inform the processing of another. Specifically, the results obtained from one technique, such as the generation of item ratings or classifications, serve as input for subsequent recommendation techniques.
   For example, consider a scenario in which a content-based recommendation system employs item feature analysis to produce ratings or classifications for items. These ratings or classifications can then be integrated into the processing of a collaborative filtering technique. Here, the collaborative filtering approach may further refine recommendations based on user interactions or preferences, incorporating the insights derived from the content-based analysis.
   This sequential approach enables each recommendation technique to build upon the insights gleaned from the previous one, potentially leading to enhanced recommendation accuracy and personalization.

7. **Meta-Level Hybrid System:** The meta-level hybrid approach involves using the model generated by one recommendation technique as input for another. Unlike feature augmentation, where learnt models generate features, in a metalevel hybrid, the entire model serves as input. For example, the web filtering system Fab employs user-specific selection agents performing content-based filtering using Rocchio's method to maintain term vector models, shared across users for document collection. Pazzani [86] describes 'content collaboration', where content-based models are built for each user and compared to make predictions. This method benefits hybrid systems by providing a compressed representation of user interests, facilitating collaborative mechanisms to operate more efficiently on dense information representations.

1.3.4 Context-Aware Recommender Systems

Context Aware Recommender Systems expand conventional recommendation frameworks by integrating contextual factors into user-item interactions. This enhancement introduces a complex multidimensional search space that presents computational complexities. The primary challenge lies in effectively capturing user preferences across diverse contextual scenarios and identifying pertinent contextual attributes for informed recommendations [69].

Types of Contextual Information

Context, defined as any information that identifies situations related to interactions between humans, applications, and their environment, exerts a substantial influence on user preferences. These contextual elements can exhibit static or dynamic characteristics and may vary in terms of observability, thereby impacting the recommendation process differently [69].
According to Andreas [128], there are five types of contextual data:

- Individuality
- Activity
- Relations
- Time

- Location

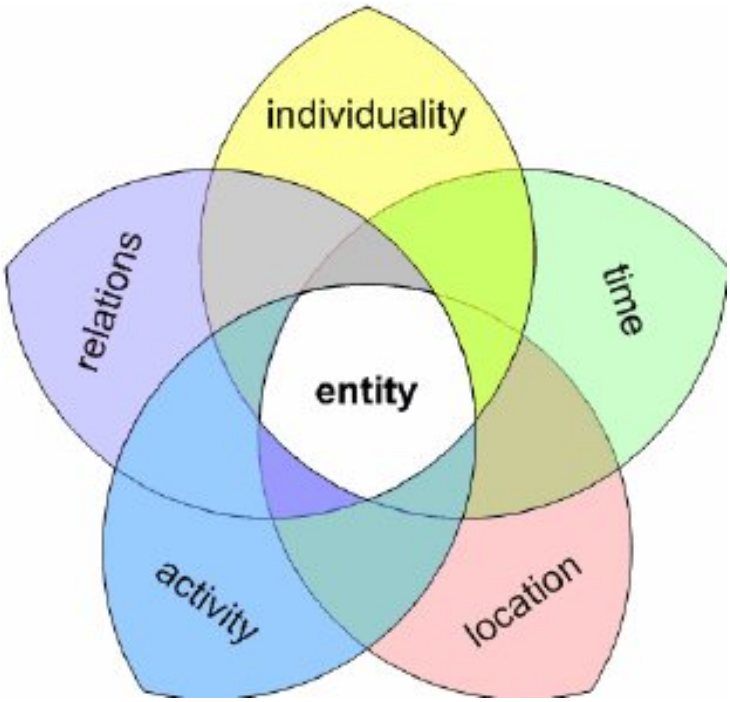

Figure 1.10: Types of Contextual Data [128].

Types of Context-Aware Recommenders

Pre-filtering: Contextual pre-filtering refers to the process of using contextual attributes to filter and select relevant ratings before they are utilized in conventional user-item recommendation systems. Taking into account contextual information such as time, location, or user preferences, irrelevant ratings are excluded, improving the accuracy and relevance of the recommendations provided to users [69].
Post-filtering: Contextual post-filtering is a method where predicted ratings are initially generated, and then contextual information is utilised to adjust these ratings for each user. This adjustment process helps refine the recommendations based on the specific context of the user. There are two main types of contextual post-filtering approaches: heuristic-based and model-based.
Contextual Modelling Approaches:

- CGR, or Graph-Based Relevance Measure: This approach, proposed by [120], evaluates a target user's relevance to a group of objects in order to incorporate contextual information into the recommendation system[50].
- Contextual-neighbors: This method, which was put out by [84], uses contextual information to calculate the level of neighborhood in a user-centric collaborative filtering configuration. Four variants of this strategy are shown by the authors, each with a different approach to contextual neighborhood selection [50].

1.4 Limitations of Recommender Systems

1.4.1 Data Sparsity Problem

In recommendation systems, the data sparsity problem is the state in which there is not enough information, such as ratings or user-item interactions, to generate reliable forecasts or suggestions for every user or item [15]. This issue appears when the overall number of users or objects in the system is far greater than the number of ratings or user interactions, leading to sparse matrices or datasets.

1.4.2 Cold Start Problem

The cold-start problem is a significant challenge in recommender systems, and it can be categorized into two main types:

1. **User's Cold-start Problem**: This occurs when a user's preferences and likings are either unknown or new to the system. Essentially, the system lacks sufficient data about the user's interests to provide accurate recommendations.

2. **Item's Cold-start Problem**: This problem arises when a new item, such as an article, is added to the recommender system, but it has no ratings or viewership data associated with it. Consequently, the system lacks information about the item's popularity or relevance to users.

These cold start challenges make it difficult for recommender systems to provide relevant and personalised recommendations whenStudy and project realization faced with limited user or item data [109].

1.4.3 Scalability

Scalability refers to the ability of a system to handle increasing demands or growth without sacrificing performance or reliability. In the context of recommender systems, scalability issues arise when the system struggles to manage larger volumes of users and items while still providing recommendations in a timely manner. This failure to cope with increased user and item load can result in longer response times, impacting the overall user experience [104].

1.4.4 Privacy and Security Issues

Privacy concerns in recommender systems are well documented. Users are often wary of using their personal information, browsing history, and preferences for recommendation purposes. There is a growing risk of data breaches or unauthorised access to sensitive user data, which can lead to privacy violations or identity theft [78].

However, security issues pose significant threats to recommender systems. Malicious attacks, such as data breaches, malware injections, phishing attacks, or denial-of-service attacks, can compromise the integrity or availability of the system. These vulnerabilities may expose user data to unauthorized access or manipulation, undermining the trustworthiness of recommendations and disrupting the system's normal functioning [68].

1.5 Review of Techniques to Address Sparsity in Recommender Systems

Recommender systems have become indispensable tools in today's e-commerce landscape, assisting users in navigating through a vast array of choices. However, their effectiveness is often hampered by the sparsity of user-item interaction data. This review provides an overview of the recent literature that addresses this issue. Innovative collaborative filtering methods, such as those based on Map-Reduce clustering, have shown enhanced performance over traditional techniques [65]. These methods leverage collective preferences to provide recommendations, even in data-scarce scenarios. Deep learning techniques, integrated with collaborative filtering, demonstrate the capacity to handle non-linear data effectively [76], allowing for more robust utilization of sparse data for recommendation purposes. Clustering bipartite networks and hybrid deep neural networks aim to address sparsity and interpretability challenges [123, 45], offering approaches to mitigate sparsity by grouping similar users or items. Techniques such as the Sequence and

Set Similarity Measure (S3M) together with Singular Value Decomposition (SVD) and novel preprocessing methods help calculate user similarity and improve the quality of recommendation [60, 93], contributing to enrichment of recommendation processes through additional similarity measures and data preprocessing. Approaches like ZeroMat and deep learning tailored for collaborative recommender systems (DLCRS) showcase superior performance [116, 16], providing methods to learn robust representations from sparse data. Using Linked Open Data (LOD) and cross-domain recommendation techniques also contribute to resolving sparsity and cold start issues [81, 124], enriching recommendations with external knowledge and data from related domains. Techniques like deep transfer learning with multimodal embedding and matrix regeneration with item features demonstrate substantial improvements in recommendation accuracy [59, 29], augmenting sparse data to enhance recommendation quality. Ensemble-based context-aware recommender systems and Fast Clustering-based Recommendation methods effectively mitigate sparsity and cold-start problems [34, 37], leveraging multiple models or clustering techniques to enhance recommendation accuracy in data-sparse scenarios. Furthermore, co-SVD for cold start recommendations and DotMat algorithms show competitive results [79, 117], mitigating the impact of sparsity on recommendation quality through imputation techniques or the use of additional information. Finally, domain-independent learning representations through shared weight auto-encoders demonstrate promising outcomes [118], enabling more accurate recommendations, even with sparse data, by capturing the underlying patterns in the data.

1.6 Evaluation Metrics for Recommender Systems

1.6.1 Mean Absolute Error (MAE)

A popular statistic in predictive modeling for assessing prediction accuracy is Mean Absolute Error (MAE). The MAE offers a clear indication of the average amount of errors between projected and actual values, in contrast to other metrics such as the RMSE. It is calculated by averaging the absolute deviations between the values that were anticipated and those that were observed. Because the absolute function guarantees that all differences are positive [100].

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_1 - \hat{y}_i|^2 \tag{1.14}$$

1.6.2 Root Mean Squared Error (RMSE)

The Root Mean Squared Error (RMSE) is a widely used metric in predictive modeling for assessing the accuracy of numerical forecasts. It is calculated as the square root of the mean of the squared differences between expected and actual values, as shown by the formula:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(S_i - O_i)^2} \tag{1.15}$$

where O_i represents the observed values, S_i signifies the predicted values, and n denotes the total number of observations considered for analysis. RMSE is favored due to its ability to offer a comprehensive evaluation of prediction accuracy across various models or setups. However, it's important to note that RMSE comparisons are only meaningful when applied to the same variable, as the interpretation of the metric is influenced by the measurement scale. [31]

1.6.3 Precision

$$\text{Precision (pr)} = \frac{\text{True Positives (tp)}}{\text{True Positives (tp)} + \text{False Positives (fp)}} \tag{1.16}$$

Precision represents the ratio of true positives to the sum of true positives and false positives which means measuring the proportion of correctly retrieved instances among all instances that were retrieved. A value of 1 signifies perfect performance with no mistakes, while a value of 0 indicates no correct detections at all. [94].

1.6.4 Recall

$$\text{Recall (r)} = \frac{\text{True Positives (tp)}}{\text{Total True (tt)}} \tag{1.17}$$

Recall represents the ratio of true positives to the total number of target signals. A value of 1 indicates that all target calls were correctly detected, while a value of 0 indicates that no single target call was correctly detected.

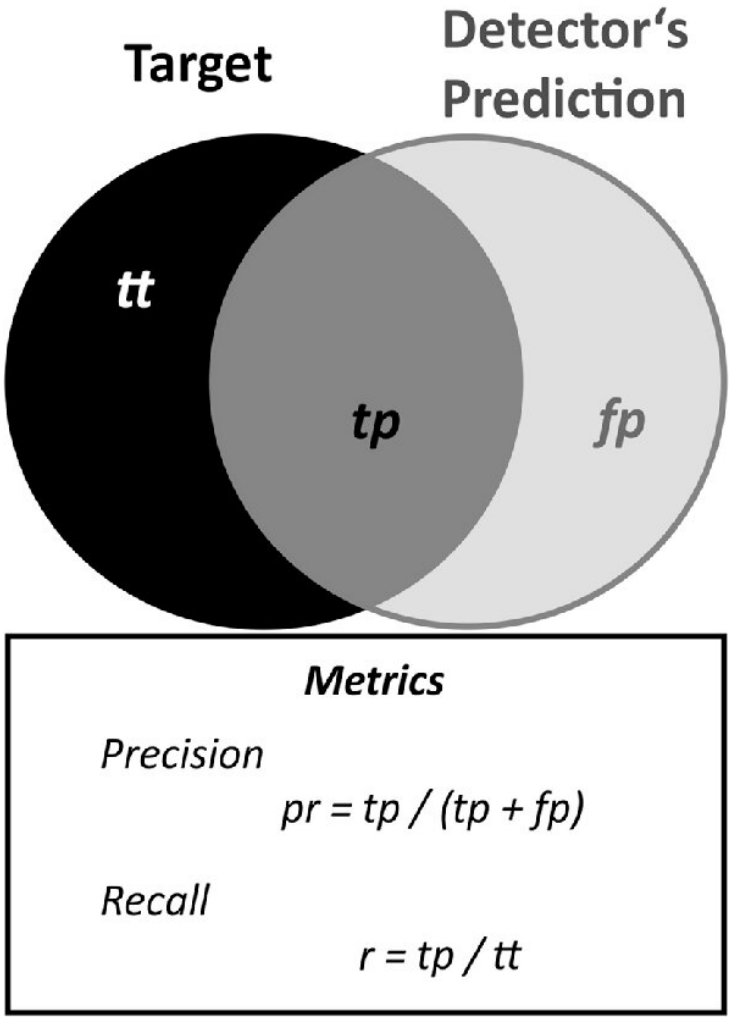

Figure 1.11: Precison And Recall graphical representation [94].

1.6.5 F1-Score

The F1-score is a metric that blends precision and recall, providing a balanced assessment of a classification model's performance, particularly useful for handling imbalanced datasets. Precision gauges the accuracy of positive predictions relative to all positive predictions, while recall measures the ability of the model to capture all actual positive instances in the dataset. Calculated as the harmonic mean of precision and recall, the F1 score offers a comprehensive evaluation, considering both false positives and false negatives. It's expressed by the formula:

$$\text{F1 score} = 2\frac{\text{PrecisionRecall}}{\text{Precision+Recall}} \tag{1.18}$$

A high F1-score signifies strong performance in correctly identifying positive instances while minimizing false positives and false negatives, making it a valuable metric for assessing classification models [92].

1.7 Sector-Specific Applications of Recommender Systems

1.7.1 E-Commerce

- **Amazon:** Amazon is a multinational technology company based in Seattle, Washington, USA. Founded by Jeff Bezos in 1994, Amazon started as an online marketplace for books, but quickly expanded into various other product categories, including electronics, apparel, and household goods. Today, Amazon is one of the largest e-Commerce platforms globally, offering a wide range of products and services, including Amazon Prime, Amazon Web Services (AWS), and Kindle e-readers [21].

- **eBay:** eBay is an online marketplace that enables individuals and businesses to buy and sell a variety of goods and services globally. Established in 1995 by Pierre Omidyar, eBay operates as a peer-to-peer platform, allowing users to list items for sale, bid on auctions, or purchase items at fixed prices. With millions of active users worldwide, eBay facilitates transactions across numerous categories, including electronics, fashion, collectibles, and automotive [40].

- **Alibaba:** Alibaba Group Holding Limited is a Chinese multinational conglomerate that specialises in e-Commerce, retail, the Internet and technology. Founded by Jack Ma in 1999, Alibaba operates various online marketplaces, including Taobao, Tmall, and Alibaba.com, which connect buyers and sellers domestically and internationally. Alibaba also provides cloud computing services, digital payment solutions, and logistics services, establishing itself as a prominent player in the global digital economy [62].

1.7.2 Streaming Services

- **Netflix:** Netflix is a subscription-based streaming service offering a wide variety of television shows, movies, documentaries, and original content. Founded in 1997 by Reed Hastings and Marc Randolph, Netflix initially operated as a DVD rental service before transitioning to a streaming platform in 2007. Today, Netflix is one of the leading providers of on-demand entertainment, with a vast library of content available to subscribers on various devices [82].

- **Hulu:** Hulu is an American subscription video-on-demand service offering a diverse selection of TV shows, movies, and original programming. Launched in 2007, Hulu provides both ad-supported and ad-free subscription plans, allowing users to stream content from major networks, cable channels, and streaming studios. Hulu also offers live TV streaming options, making it a comprehensive entertainment platform for viewers [57].

- **Disney+:** Disney+ is a subscription-based streaming service owned and operated by The Walt Disney Company. Launched in 2019, Disney+ offers a vast library of Disney, Pixar, Marvel, Star Wars, and National Geographic content, including movies, TV shows, and exclusive original series. With a focus on family-friendly entertainment, Disney+ has quickly become a popular choice for fans of Disney's iconic characters and franchises [111].

- **Spotify:** Spotify is a digital music streaming service that provides access to millions of songs, podcasts, and audiobooks from various artists and creators around the world. Founded in Sweden in 2006, Spotify offers free and premium subscription tiers, allowing users to listen to music on demand, create playlists, and discover new tracks. With its user-friendly interface and extensive music catalogue, Spotify has become one of the leading platforms for streaming music globally [107].

1.7.3 Social Media

- **Meta:** formerly Facebook, has been at the forefront of revolutionizing human connection since its establishment in 2004. The company has made billions of users worldwide possible with platforms such as Instagram, WhatsApp, and Messenger by providing state-of-the-art technology that fosters community growth, communication, and business progress. In addition to traditional 2D screens,it is now embracing immersive experiences such as virtual reality (VR) and augmented reality (AR) due to the rapid growth of technology. The company aims to spearhead the next social technology revolution by exploring these immersive technologies, offering consumers ever more engaging and immersive ways to engage and communicate [77].

1.7.4 Online News and Publishing

1. **Apple News:** Apple News is a news aggregation platform launched by Apple in 2015, reaching over 1 billion iOS devices. With 85 million monthly active users, it fills the gap left by changes in Facebook's algorithms. Publishers can create channels and publish articles using various formats. The app features sections like "Today" and "News+" for headlines and magazine content, respectively. Human editors curate the "Top Stories" section, while algorithms determine "Trending Stories." This combination shapes the content users see, influencing news consumption patterns [18].

2. **Google News:** Google News is a news aggregator service developed by Google. It was launched in 2002 and is available on the Web and as a mobile app for iOS and Android devices. Google News uses artificial intelligence algorithms to analyse and organise news articles from thousands of sources around the world. It presents users with personalized news feeds based on their interests, location, and search history. Google News also offers features such as topic-based navigation, fact-checking labels, and coverage timelines to help users explore and understand current events [1].

1.7.5 Education and Online Learning

- **Coursera:** Founded in 2012 by Stanford professors Andrew Ng and Daphne Koller, is a leading online learning platform with 148 million registered learners as of March 31, 2024. It partners with over 325 universities and industry leaders to offer a wide range of courses, certifications, and degree programs. Coursera's mission is to provide accessible, high-quality education worldwide, empowering people and institutions to improve and improve skills in fields such as data science, technology, and business [32].

- **Udemy:** A renowned global learning platform facilitating skill acquisition and development for organizations and individuals. With an extensive marketplace and over 69 million learners worldwide, Udemy connects thousands of instructors with diverse content offerings. Using advanced technology, such as artificial intelligence, Udemy delivers personalised learning experiences. Their enterprise solution, Udemy Business, offers curated content for companies, enabling on-demand learning, specialised training, and leadership development programmes [114].

1.8 Conclusion

In this chapter, we did a thorough overview of recommender systems, starting from their foundational concepts to the latest advancements in the field. We began by understanding the importance and evolution of these systems, recognising their pivotal role in modern information retrieval and decision-making processes.

Exploring the various types of recommender systems, including collaborative filtering, content-based filtering, hybrid systems, and context-aware recommender systems, allowed us to grasp the diverse approaches used to generate recommendations. Each type was dissected, highlighting its mechanisms, techniques, and classification, offering readers a comprehensive understanding of their inner workings.

However, we also acknowledged the limitations inherent in these systems, such as the data sparsity problem, cold start problem, scalability issues, and privacy concerns. Addressing these challenges is crucial for enhancing the effectiveness and reliability of recommender systems in real-world applications.

The chapter further delved into the review of techniques aimed at mitigating the data sparsity problem, laying the groundwork for improving the performance of recommendation algorithms. Additionally, we discussed the evaluation metrics used to assess the performance of recommender systems, providing readers with essential tools for measuring and comparing different algorithms accurately.

Lastly, we explore sector-specific applications of recommender systems. By examining real-world use cases, we gained valuable insight into how recommender systems are used to enhance user experience, drive business growth, and revolutionise various industries.

Chapter 2

Conceptual Study

2.1 Introduction

In the rapidly evolving digital marketplace, e-commerce platforms strive to offer personalised experiences to their users through recommender systems. These systems aim to predict and suggest products that align with user preferences and behaviors, enhancing user satisfaction and driving sales. However, despite their potential, recommender systems face significant challenges, with one of the most pressing being the sparsity problem. Sparsity occurs when the user-item interaction matrix, which underlies recommendation algorithms, is predominantly empty because users typically interact with only a small subset of available items. This lack of sufficient interaction data limits the system's ability to accurately learn and predict user preferences, adversely affecting the quality of recommendations and overall user experience.

Addressing the sparsity problem is crucial for improving the effectiveness of recommender systems. In this study, we propose a novel approach to tackle this issue by integrating Latent Dirichlet Allocation (LDA), an autoencoder, and a Convolutional Neural Network (CNN). LDA is used to extract latent topics from user-item interactions, providing a richer representation of user preferences and item characteristics. The autoencoder then learns compact and informative representations of the data, reducing dimensionality while preserving essential features. Finally, the CNN captures complex patterns and interactions within the data, enhancing the accuracy and reliability of the recommendations. This combined approach aims to overcome the limitations posed by data sparsity and deliver more personalized and relevant product suggestions.

2.2 Motivation and Objective

Recommender systems have become essential tools for filtering the vast amount of information available on the internet. However, the issue of data sparsity, where user-item interactions are scarce, poses a significant challenge and reduces the effectiveness of these systems. This scarcity limits the ability of recommender systems to accurately predict user preferences, thus affecting their performance. Addressing this challenge is crucial for improving the reliability and relevance of recommendations in various applications, particularly in e-Commerce.

The primary objective of this project is to build a content-based recommender system that addresses the problem of sparsity in the domain of e-commerce. Our approach involves leveraging advanced techniques in topic modelling and neural networks through the following steps:

Firstly, we utilise Latent Dirichlet Allocation (LDA) to represent items by identifying their topic distributions. These topic representations are then encoded using a Multilayer Perceptron (MLP) autoencoder to compress and capture the essential features of the items. Finally, we train a generative Convolutional Neural Network (CNN) model with the encoded items to learn complex patterns in user-item interactions.

This multi-step approach aims to harness the complementary strengths of each technique to enrich data representation and improve recommendation quality. Specifically, our objectives include the following.

- **Extracting Rich Features**: Using LDA to obtain a semantic understanding of items and an MLP autoencoder to efficiently compress this information.
- **Training a Robust Model**: Leveraging a CNN to capture interaction patterns and generate high-quality recommendations.
- **Evaluating System Effectiveness**: Assessing the hybrid model using standard metrics such as precision, recall, Mean Squared Error (MSE), and Mean Absolute Error (MAE) to ensure its effectiveness compared to traditional methods.

By integrating these techniques, we aim to overcome the limitations of existing solutions and develop a recommender system capable of providing accurate and relevant recommendations, even in the presence of sparse data.

2.2.1 Problem Statement

Recommender systems often struggle with the issue of data sparsity, which occurs when the available data on user-item interactions are limited and sparse. This problem leads to several challenges:

- **Limited Data**: The lack of sufficient user-item interactions results in an incomplete understanding of user preferences, making it difficult to generate accurate recommendations.

- **Poor Generalisation**: Traditional recommendation algorithms may not generalize well to sparse datasets, leading to suboptimal performance.

Traditional recommendation methods, such as matrix factorization or neighborhood-based approaches, struggle with data sparsity. They often fail to capture the deep semantic connections in sparse interactions, leading to less accurate recommendations. Therefore, developing solutions that can better interpret the latent relationships between users and items is crucial. LDA can uncover underlying topics within content, while CNNs are adept at identifying complex features in interactions.

2.2.2 Hypotheses

- **Hypothesis 1**: Integrating LDA-based topic modelling into recommender systems can enhance the understanding of user preferences by identifying relevant topics in the content of the items.
  - *Justification*: LDA is proficient at extracting latent themes from extensive text corpora, which can provide richer data for making recommendations.
- **Hypothesis 2**: Utilising Convolutional Neural Networks (CNNs) improves the ability of recommender systems to detect intricate patterns in user-item interactions, even when data is sparse.
  - *Justification*: CNNs are effective at recognizing complex and hierarchical patterns in data, which is particularly beneficial in addressing the sparsity issue.
- **Hypothesis 3**: Combining LDA and CNNs in a hybrid model can mitigate the limitations of traditional methods by offering a more comprehensive representation of user-item interactions.
  - *Justification*: The hybrid approach leverages the strengths of both methods to provide more accurate and relevant recommendations.

The current chapter will detail the methodology used to implement and test our approach. It will cover the steps involved in data preprocessing, the configuration of LDA and CNN models, and the evaluation criteria for our hybrid recommender system.

2.3 Design and Architecture

Introducing the design of our recommender system aimed at addressing the issue of data sparsity in e-commerce. The system leverages advanced techniques in topic modelling and neural networks to provide accurate and relevant recommendations. The architecture of the system is illustrated in Figure 2.1:

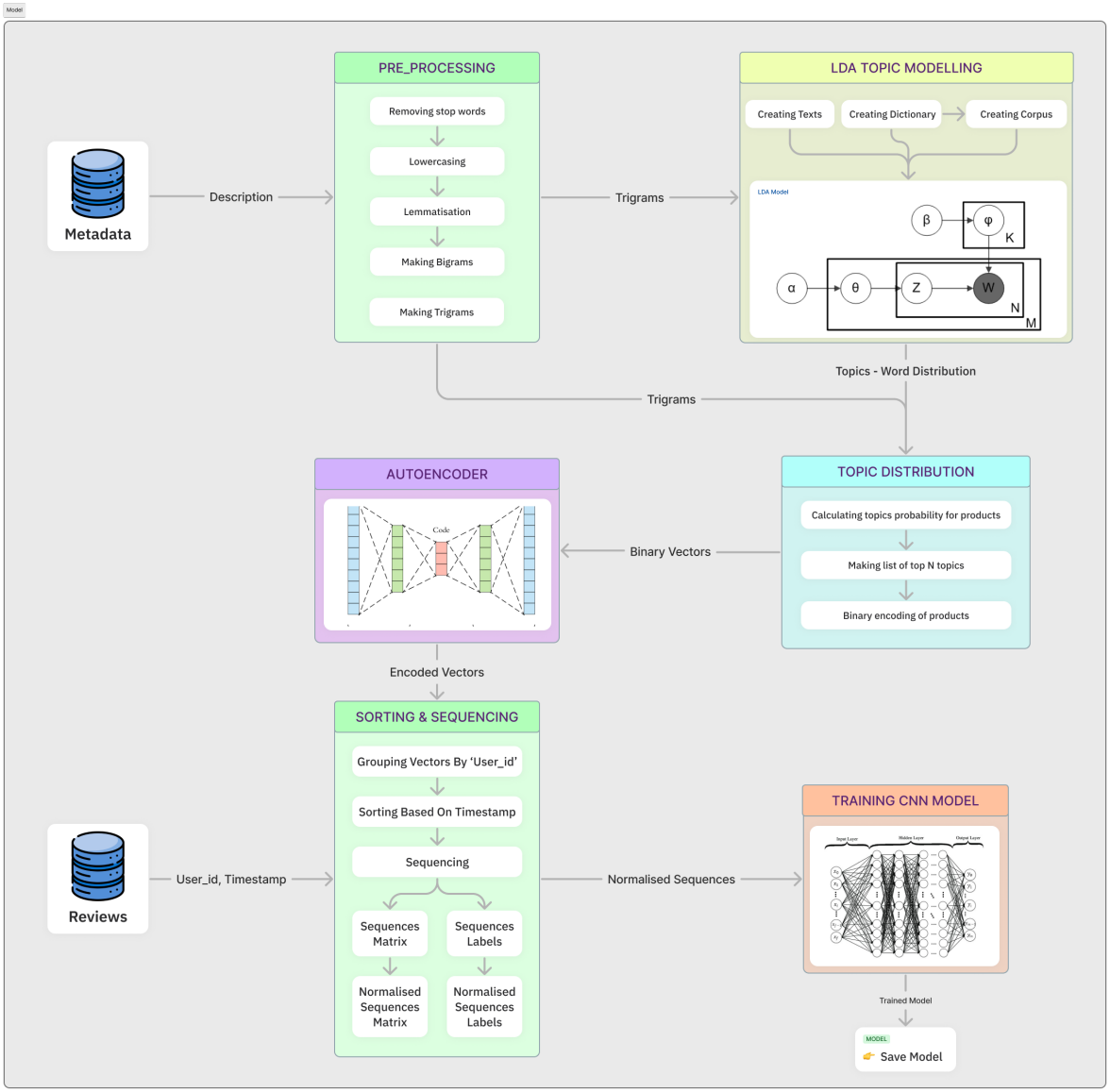

Figure 2.1: Illustration of the system's architecture .

2.4 Dataset: Amazon Baby Products

The dataset consists of Amazon Baby Products data from Amazon-Reviews-2023 [55], which includes extensive metadata and reviews for millions of baby products. The dataset details are as follows:

Reviews Columns

Field	Type	Explanation
rating	float	Rating of the product (from 1.0 to 5.0).
title	str	Title of the user review.
text	str	Text body of the user review.
images	list	Images that users post after they have received the product. Each image has different sizes (small, medium, large), represented by the small_image_url, medium_image_url, and large_image_url respectively.
asin	str	ID of the product.
parent_asin	str	Parent ID of the product. Note: Products with different colors, styles, sizes usually belong to the same parent ID. The "asin" in previous Amazon datasets is actually parent ID. Please use the parent ID to find the product meta.
user_id	str	ID of the reviewer
timestamp	int	Time of the review (unix time)
verified_purchase	bool	User purchase verification
helpful_vote	int	Helpful votes of the review

Table 2.1: User Reviews Data -Amazon reviews 2023-

Metadata Columns

Field	Type	Explanation
main_category	str	Main category (i.e., domain) of the product.
title	str	Name of the product.
average_rating	float	Rating of the product shown on the product page.
rating_number	int	Number of ratings in the product.
features	list	Bullet-point format features of the product.
description	list	Description of the product.
price	float	Price in US dollars (at time of crawling).
images	list	Images of the product. Each image has different sizes (thumb, large, hi_res). The "variant" field shows the position of the image.
videos	list	Videos of the product including title and url.
store	str	Store name of the product.
categories	list	Hierarchical categories of the product.
details	dict	Product details, including materials, brand, sizes, etc.
parent_asin	str	Parent ID of the product.
bought_together	list	Recommended bundles from the websites.

Table 2.2: Product Metadata -Amazon reviews 2023-

This rich dataset provides ample information for our recommendation system, though it also presents challenges due to its size and sparsity.

2.5 Applied Methods

In this section, we will outline the fundamental techniques and strategies employed to enhance recommendation accuracy and address the issue of sparse user-item interactions. By integrating advanced methodologies, our goal is to develop a robust and effective recommender system capable of delivering accurate and relevant recommendations.

2.5.1 Data Preprocessing

In this section, we describe the preprocessing steps applied to the dataset to prepare it for topic modelling and recommendation tasks.

Text Preprocessing

The first step in preprocessing involves cleaning and normalising the product descriptions. This is crucial to ensure that the textual data is in a consistent format and free of noise, which can improve the quality of topic modeling and subsequent analysis.

We perform the following operations using the spaCy library:

- **Character Removal**: Remove all characters except letters and spaces to eliminate punctuation, numbers, and other non-alphabetic symbols.
- **Lowercasing**: Convert all text to lowercase to maintain uniformity.
- **Tokenisation**: Split the text into individual tokens (words).
- **Stop Word Removal**: Remove common stop words (e.g., "the", "and") that do not carry significant meaning.
- **Lemmatiaation**: Convert words to their base or root form (e.g., "running" to "run") to reduce the dimensionality of the text data.

The preprocessing function is implemented as follows:

```
# Load spaCy's English tokeniser, stopwords, and lemmatizer
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])

def preprocess(text):
    # Remove all characters except letters and spaces
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    # Convert it to lowercase
    text = text.lower()
    # Tokenize, remove stop words, and lemmatize using spaCy
    tokens = [token.lemma_ for token in nlp(text) if not token.is_stop]
    return ' '.join(tokens)


md['desc_prcssd'] = md['description'].apply(preprocess)
```

After preprocessing, rows with empty `desc_prcssd` columns are removed to ensure that only meaningful text data are retained.

Extract Relevant Columns

Next, we extract the `parent_asin` and `desc_prcssd` columns and create a new DataFrame. This step isolates the essential information required for topic modelling and subsequent recommendation tasks.

```
# Select relevant columns
md_filtered = md[['parent_asin', 'desc_prcssd']]

# Display the first few rows of the filtered DataFrame
print(md_filtered.head())

# Output:
#   parent_asin                                        desc_prcssd
# 0  B01C4319LO  product description ultimate convenience chicc...
# 1  B07FM4MJJP  choose big size confuse size medium age mont...
# 2  B0083SXABC  baby begin interest feed have right equipment ...
# 3  B07N8GRHHK  mdesign clear storage bin kids supply nursery ...
# 4  B00ITJD0U6  kandoo beauty product design care pamper feel ...
```

The resulting shape of the filtered DataFrame is `(126602, 2)`, indicating that we have 126,602 entries with the relevant information.

Generate Bigrams and Trigrams

The process of enhancing the semantic understanding of textual data in natural language processing tasks often entails the construction of higher-order linguistic models, such as bigrams and trigrams. In this study, we employed Gensim's Phrases class, a widely utilised tool in the field, to facilitate the creation of such models.

First, we initialized the spaCy language model to tokenize the text data:

```
nlp = spacy.load("en_core_web_sm")
md_filtered['tokenized_text'] =
md_filtered['desc_prcssd'].apply(lambda x: [token.text for token in nlp(x)])
```

Next, a bigram model was constructed using the `Phrases` class, wherein pairs of words exhibiting notable frequency and co-occurrence within the corpus were identified. This process, governed by parameters such as `min_count` and `threshold`, ensured the formation of meaningful word pairs, enhancing the granularity of the linguistic representation:

```
bigram_words =
gensim.models.Phrases(md_filtered['tokenized_text'], min_count=5, threshold=50)
```

The `min_count` parameter determines the minimum frequency a phrase must have in the corpus to be considered for inclusion in the model. It serves as a filter to exclude infrequent word combinations that may introduce noise or outliers into the model. By setting an appropriate `min_count`, we ensure that only sufficiently common phrases contribute to the linguistic representation, thus controlling the granularity of the model.

On the other hand, the `threshold` parameter governs the strength of association required for two words to form a phrase. It controls the threshold for forming phrases based on their co-occurrence statistics within the corpus. A higher threshold value imposes a stricter criterion for identifying phrases, resulting in fewer but more salient linguistic associations. Conversely, a lower threshold value allows for the inclusion of a broader range of word combinations, potentially capturing more diverse linguistic patterns.

Subsequently, the bigram model served as the foundation for the development of a trigram model. By applying the `Phrases` class on the output of the bigram model, sequences of three words demonstrating significant association were identified:

```
trigram_words =
gensim.models.Phrases(bigram_words[md_filtered['tokenized_text']], threshold=50)
```

Following the construction of these models, a pivotal step involved their conversion into Phrasers, more memory-efficient variants tailored for text transformation tasks. This conversion facilitated seamless integration of the models into the text processing pipeline, ensuring optimal performance and resource utilisation:

```
bigram_mod = gensim.models.phrases.Phraser(bigram_words)
trigram_mod = gensim.models.phrases.Phraser(trigram_words)
```

Thus equipped with the bigram and trigram Phrasers, we were poised to augment the semantic richness of the text through their application. By replacing common word pairs and triplets with their respective bigram and trigram representations, we sought to unveil deeper linguistic insights embedded within the corpus, thereby advancing our understanding of the underlying semantic structures:

```
md_filtered['bigrams'] = md_filtered['tokenized_text'].apply(lambda x: bigram_mod[x])
md_filtered['trigrams'] = md_filtered['bigrams'].apply(lambda x: trigram_mod[x])
```

2.5.2 LDA Topic Modeling

This part is delve into the steps needed for LDA Topic Modelling.

Preparing the LDA model input

To prepare the input for the LDA model, it is necessary to create a dictionary and a corpus from the processed text. The dictionary maps each word to a unique ID, while the corpus represents each document as a bag-of-words, which is a list of tuples indicating the frequency of each word in the document.

First, the dictionary is created from the trigrams and the texts also:

```
id2word = corpora.Dictionary(md_filtered['trigrams'])
texts = md_filtered['trigrams']
```

next, the corpus is created, which is a list of term-document frequency representations for each document in the texts:

```
corpus = [id2word.doc2bow(text) for text in texts]
```

In this context:

- `id2word` is a dictionary object that maps each unique word in the corpus to a unique ID. - `texts` is the collection of processed documents (trigrams) from which the LDA model will learn. - `corpus` is the term-document frequency representation of the texts, which serves as the input for the LDA model.

The dictionary and corpus form the essential inputs for training the LDA model, enabling it to identify the latent topics within the text data.

Defining the LDA model

We define a function to calculate the coherence score of the LDA model. Coherence score is a metric used to evaluate the quality of the topics generated by the model. Measures the degree of semantic similarity between high-scoring words in the topic.

```
lda_model = LdaMulticore(corpus=corpus,
             id2word=id2word,
             num_topics=n,
             random_state=100,
             chunksize=100,
             passes=10,
             alpha=alpha,
             per_word_topics=True,
             eta=beta)
```

The provided code snippet initializes an LDA (Latent Dirichlet Allocation) model using the `LdaMulticore` class, a variant of LDA designed for multicore processors. Latent Dirichlet Allocation is a widely-used probabilistic model for uncovering the underlying thematic structure in a collection of documents.

In the context of natural language processing, LDA assumes that each document in the corpus is generated from a mixture of topics, and each word in the document is attributed to one of these topics. The goal of LDA is to infer these latent topics based on the observed word frequencies in the documents.

The parameters of the LDA model play a crucial role in shaping its behaviour and performance:

- **corpus**: This parameter represents the document-term matrix, where each row corresponds to a document and each column corresponds to a unique term in the vocabulary. It encapsulates the textual data on which the LDA model is trained.
- **id2word**: This parameter is a mapping from word IDs to words in the vocabulary. It allows the LDA model to interpret the document-term matrix in human-readable terms.
- **num_topics**: Specifies the number of topics to be inferred from the documents. This parameter is essential as it determines the granularity of the topics discovered by the model.
- **random_state**: Seed for the random number generator. Setting a seed ensures reproducibility of results across different runs of the model.
- **chunksize**: The number of documents to be processed in each chunk during training. Larger chunk sizes may lead to faster training but may require more memory.

- **passes**: The number of passes over the entire corpus during training. Each pass involves updating the topic assignments for each word in each document based on the current state of the model.
- **alpha**: The parameter controlling the document-topic density. A higher alpha value results in documents being composed of more topics, while a lower alpha value results in documents being composed of fewer topics.
- **per_word_topics**: If True, the model returns the per-word topic assignments along with the final topics.
- **eta**: The parameter controlling the topic-word density. Similar to alpha, a higher eta value results in topics being composed of more words, while a lower eta value results in topics being composed of fewer words.

By adjusting these parameters, researchers can fine-tune the LDA model to better capture the underlying thematic structure of the documents and generate more accurate topic assignments.

Training and Evaluating the LDA Model

This stage involves the training and subsequent evaluation of the Latent Dirichlet Allocation (LDA) model, a probabilistic approach widely used for topic modeling in natural language processing tasks.

The training of the LDA model is conducted with meticulous attention to predefined parameters, including the number of topics (`n`), the Dirichlet prior for the document-topic distribution (`alpha`), and the Dirichlet prior for the topic-word distribution (`beta`). Following model training, a rigorous evaluation process ensues, where the efficacy of the model is gauged through two key metrics: **perplexity** and **coherence score**. **Perplexity** serves as a measure of the model's ability to predict unseen data, reflecting its capacity to capture the underlying structure of the corpus. A lower perplexity score indicates a higher degree of model coherence and predictive accuracy.

Concomitantly, the **coherence score**, derived from semantic similarity measures between top-scoring words within topics, provides an assessment of the interpretability and semantic coherence of the generated topics. A higher coherence score signifies more coherent and interpretable topics, indicative of a more robust and representative model.

Through the meticulous evaluation of these metrics, researchers gain critical insights into the performance and effectiveness of the LDA model in uncovering latent topics within the corpus. This systematic assessment contributes to the refinement and optimization of topic modeling methodologies, fostering deeper insights into the underlying thematic structure of textual data.

2.5.3 Topic Distribution and Vector Creation

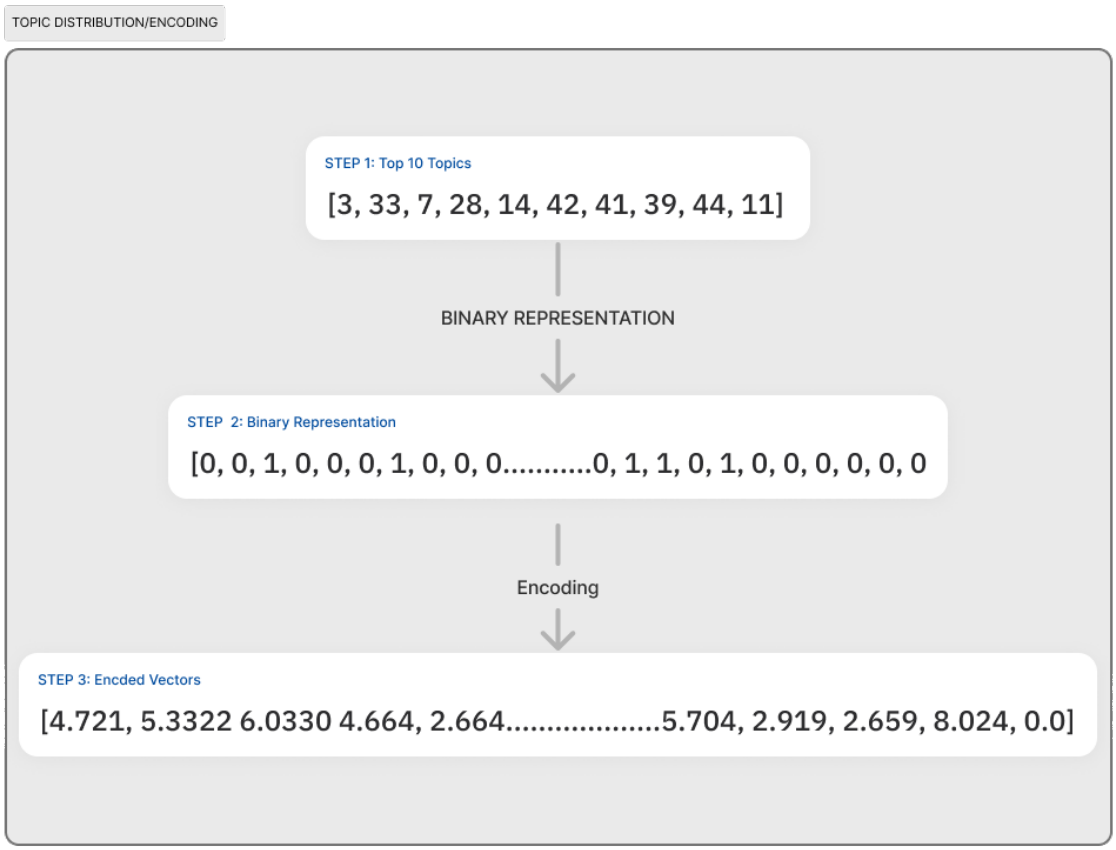

Figure 2.2: Items' vectors making .

After training the LDA model, the following steps were taken to create vectors representing the items:

Topic Assignment and Top-N Topics Extraction

The process of topic assignment and subsequent extraction of the top-N topics constitutes a pivotal phase in the analysis of textual data. Leveraging the trained LDA model, each document within the dataset undergoes a meticulous assessment to ascertain its underlying thematic composition. This involves computing the probability distribution of topics for every document, thereby assigning a nuanced representation of topics to each.

Following topic assignment, top N most relevant topics are meticulously extracted from each document. This intricate procedure entails sorting the topics based on their probabilities and selecting the highest-ranking ones. By prioritising the most salient thematic elements, this extraction process furnishes a focused representation of the underlying thematic motifs encapsulated within each document.

By integrating these two processes, a comprehensive understanding of the thematic landscape within the dataset is attained. This combined approach not only facilitates the identification of prevalent themes but also enables the extraction of the most significant thematic elements, thereby enhancing the interpretability and utility of the topic modelling outcomes.

Document-Topic Vector Creation

In the process of document-topic vector creation, binary vectors were generated for each document, representing the presence or absence of the top N topics. Each vector element was assigned a value of 1 to indicate the presence of a topic and a value of 0 to signify its absence. This binary representation was crucial for encapsulating the thematic essence of each document in a structured and interpretable format.

These steps effectively transformed the textual data into structured vectors, each representing the thematic content of the corresponding document in a succinct manner. By doing so, the complex and

nuanced information embedded in the textual data was distilled into a concise format, facilitating further computational analysis and interpretation.

Definition and Training of Autoencoders

To enhance the processing of the document-topic vectors, an autoencoder was meticulously defined and trained to learn a compressed representation of the topic distributions. This autoencoder served as a powerful tool for dimensionality reduction, enabling the distillation of essential features from the document-topic vectors while preserving their most salient information. The autoencoder architecture and training process were designed to ensure optimal performance in capturing the underlying thematic structures within the document-topic vectors.

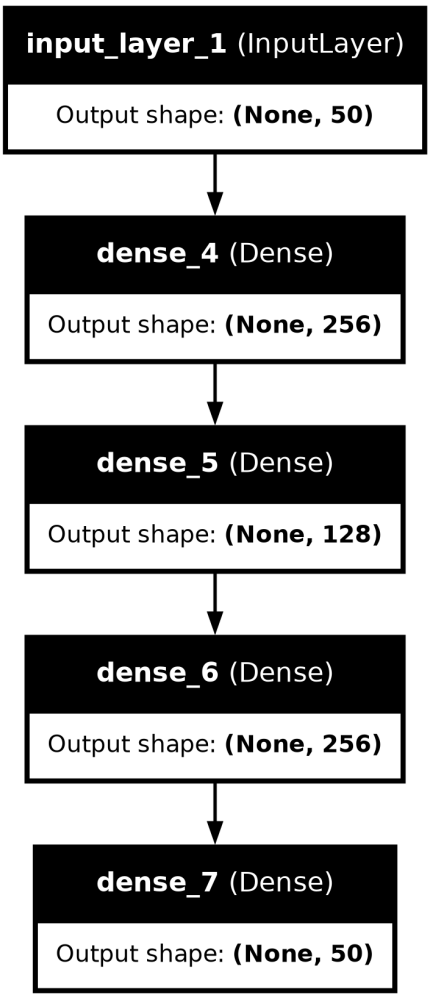

Figure 2.3: Summary of the MLP autoencoder's architecture.

The architecture included the following layers:

- **Input Layer**: The input layer's shape matched the number of topics.
- **Encoding Layers**: Two dense layers, the first with ReLU activation, and the second with ReLU activation.
- **Decoding Layers**: Two dense layers mirroring the encoding layers, each with ReLU activation, followed by an output layer with sigmoid activation.

The autoencoder was compiled with the Adam optimiser and the binary cross-entropy loss function. The Adam optimiser was selected for its efficiency and ability to handle sparse gradients, making it well-suited for training deep neural networks. The binary cross-entropy loss function was used because of its suitability for binary classification tasks, aligning with the nature of the document-topic vectors.

The training process involved running the autoencoder on the document-topic vectors for several epochs, with a specified batch size and a validation split. This training regimen was designed to ensure the model's robustness and ability to generalise well to unseen data.

Model Architecture and Parameters

The architecture of the autoencoder and its parameters are summarised in the following table.

```
Layer (type)                    Output Shape              Param #

input_layer (InputLayer)        (None, X)                 0
dense (Dense)                   (None, Y)                 X
dense_1 (Dense)                 (None, Z)                 Y
dense_2 (Dense)                 (None, Y)                 Z
dense_3 (Dense)                 (None, X)                 W
Total params: XYZ (X.Y KB)
Trainable params: XYZ (X.Y KB)
Non-trainable params: 0 (0.00 B)
```

The autoencoder architecture consists of several layers, each with a distinct role in processing the document-topic vectors. The input layer, with a shape of $(None, X)$, serves as the entry point for the vectors, where X represents the number of topics. The first dense layer, an encoding layer, transforms the input into an output shape of $(None, Y)$ and employs a ReLU activation function to introduce non-linearity, thus enabling the learning of complex patterns. The subsequent dense layer, also part of the encoding process, further refines the representation with an output shape of $(None, Z)$ and uses ReLU activation to capture intricate data structures. The first decoding layer mirrors the first encoding layer, reconstructing the encoded information into an output shape of $(None, Y)$ with ReLU activation. The final dense layer completes the decoding process, outputting a shape of $(None, X)$ and using a sigmoid activation function to map the values between 0 and 1, suitable for binary classification tasks. The total number of parameters in the model is denoted by XYZ, occupying approximately $X.Y$ KB of memory. All parameters are trainable, ensuring that the model can optimise its weights during training to achieve optimal performance. This detailed architectural overview underscores the methodical design choices that enhance the model's capacity to effectively encode and decode document-topic vectors, thereby improving the representation of topic distributions for subsequent analytical tasks.

This detailed architecture description highlights the layers and their respective roles, ensuring a comprehensive understanding of the autoencoder's construction and its capability to efficiently compress and represent the document-topic vectors. The choice of activation functions, optimiser, and loss function was critical to training the autoencoder to effectively learn the underlying patterns within the data, ensuring robust performance in subsequent tasks.

- **Relu Activation Function:** The Rectified Linear Unit (ReLU) distinguishes itself due to its simplicity and computational efficiency. Defined formally, this function acts as the identity for positive inputs and returns zero for negative inputs:

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

  Thus, ReLU yields values within the range $[0, \infty)$. For non-negative inputs, the derivative of the ReLU function is one, while it is zero otherwise. This property allows ReLU to circumvent the computational challenges posed by the Tanh and Logistic Sigmoid functions. However, ReLU is not without its drawbacks, notably the issue of vanishing gradients for negative input values. However, ReLU has found widespread adoption in various deep learning architectures. The enhancements and modifications made to the fundamental ReLU activation function will be extensively discussed in subsequent sections [39].

- **Adam Oprimiser:** Adam is a stochastic objective function optimization technique. Because it makes use of adaptive estimates of lower-order moments, it is computationally efficient and simple to apply. Adam works well with large-scale data and/or parameter problems and has modest memory needs. It works effectively in situations with noisy or sparse gradients and non-stationary

targets. The hyperparameters of the method are easily understood and need little adjustment. Its effectiveness in comparison to other stochastic optimization techniques is demonstrated by empirical results [64].

Step 5: Encoding of document-topic vectors

Post-training, the encoder part of the autoencoder was used to encode the document-topic vectors into a lower-dimensional space (128 dimensions). This process resulted in providing a compressed representation of the topic distributions for further analysis and use in recommendation tasks.

2.5.4 Data Merging and Filtering

The metadata DataFrame (`md`) and the reviews DataFrame (`rd`) were merged in the `parent_asin` column using an inner join. This ensured that only the items present in both DataFrames were retained in the merged DataFrame (`mrd`).

```
mrd = pd.merge(md, rd, on='parent_asin', how='inner')
```

Next, a new column, `binary_rating`, was created in the merged DataFrame. Ratings were converted to binary values where ratings of 3 or higher were assigned to 1 (indicating a positive rating), and ratings below 3 were mapped to 0 (indicating a negative rating). This transformation simplified the original rating scale, facilitating a more straightforward analysis.

```
# Create a new column 'rating_binary'
mrd['binary_rating'] = mrd['rating'].apply(lambda x: 1 if x >= 3 else 0)
```

To ensure the dataset contained sufficient user activity for meaningful analysis, we first determined the number of unique users by counting distinct `user_id` values. Users with fewer than 6 reviews were filtered out, retaining only those with at least 6 occurrences. This step ensured that each user in the filtered DataFrame had sufficient review data for subsequent analysis.

```
# Filter out user_ids that have fewer than 6 occurrences
user_ids_to_keep = user_counts[user_counts >= 6].index
filtered_mrd = mrd[mrd['user_id'].isin(user_ids_to_keep)]
```

Finally, the DataFrame was sorted by `user_id` and `timestamp` to arrange the reviews chronologically for each user. This chronological ordering was crucial for maintaining the temporal sequence of user interactions with items, which is essential for the subsequent sequencing and analysis steps.

```
mrd = mrd.sort_values(by=['user_id', 'timestamp'])
```

These steps resulted in a refined data set in which the review data of each user was adequately represented, binarized, and ordered, setting the stage for further analysis and model training.

2.5.5 Data Sequencing and Normalisation

To prepare the data for training the recommendation model, we performed a series of steps to sequence and normalise the encoded vectors.

Firstly, we initialised the lists to store the input sequences and their corresponding labels. A sequence length of 5 was defined, meaning each sequence would consist of 5 consecutive encoded vectors, and the label would be the vector following these 5 vectors.

For each unique user in the dataset, we filtered the data to obtain the entries corresponding to that user and extracted their encoded vectors. By iterating through each user's data, we generated sequences of vectors. Specifically, for each position in the user's data where a complete sequence could be formed (i.e., where there were at least 5 subsequent vectors), we appended the sequence of 5 vectors to our input list and the subsequent vector to our label list.

After generating sequences for all users, the lists were converted to numpy arrays. This conversion facilitated efficient processing in subsequent steps. We verified the correctness of the sequence generation process by checking the shapes of the resulting arrays.

Next, we normalised the data by scaling the input sequences and labels. Normalisation was performed by subtracting the mean from each value and then dividing by the same mean. This scaling step ensured that the data had a mean of zero and a consistent scale, which is crucial for the stability and performance of the learning algorithms used in model training.

Algorithm 1 Sequence Generation

Require: Encoded vectors dataset mrd, Sequence length $sequence\_length$
Ensure: Input sequences X, Labels y

$X \leftarrow []$ ▷ Initialize list to store sequences
$y \leftarrow []$ ▷ Initialize list to store labels
for $user\_id$ in unique IDs of mrd **do**
  $user\_data \leftarrow$ data for $user\_id$ in mrd
  $user\_vectors \leftarrow$ encoded vectors for $user\_id$
  for i in range(length of $user\_vectors - sequence\_length$) **do**
    $sequence \leftarrow user\_vectors[i : i + sequence\_length]$
    Append $sequence$ to X
    Append $user\_vectors[i + sequence\_length]$ to y
  end for
end for
Print total number of sequences generated
Convert X and y to numpy arrays

Algorithm 2 Normalization of Input Sequences and Labels

Compute the maximum and minimum values of the input sequences and labels:
max_val_X ← X.max()
min_val_X ← X.min()
max_val_y ← y.max()
min_val_y ← y.min()
Calculate the average of the maximum and minimum values:
avg_X ← (max_val_X + min_val_X)/2
avg_y ← (max_val_y + min_val_y)/2
Calculate the scaling factor:
scaling_factor_X ← (max_val_X − min_val_X)/2
scaling_factor_y ← (max_val_y − min_val_y)/2
Subtract the scaling factor from each value in the input sequences and labels:
$X\_normalized \leftarrow (X - \text{avg\_X})/\text{scaling\_factor\_X}$
$y\_normalized \leftarrow (y - \text{avg\_y})/\text{scaling\_factor\_y}$

2.5.6 Model Architecture

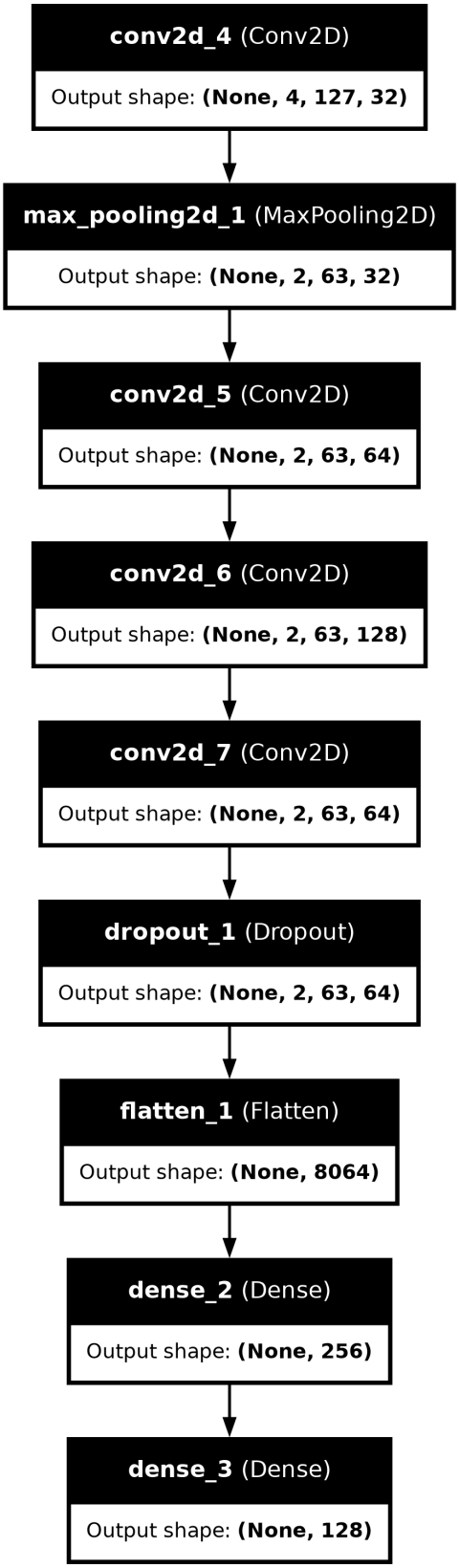

Figure 2.4: CNN model's architecture .

The convolutional neural network (CNN) model used for this project is designed to process sequences of encoded vectors. The architecture consists of several layers, each with specific functions and parameters.

Below is a detailed explanation of each layer in the model.

- **Input Layer**: The input layer is designed to accept input data with a shape of (5, 128, 1). This corresponds to sequences of 5 encoded vectors, each of length 128, with a single channel.
- **Convolutional Layer 1**: The first convolutional layer applies 32 filters with a kernel size of 2. The ReLU (Rectified Linear Unit) activation function is used to introduce non-linearity. This layer helps in detecting low-level features such as edges or textures in the input sequences.
- **Max Pooling Layer 1**: Following the first convolutional layer, a max pooling layer is used to reduce the spatial dimensions of the feature maps.
  This layer helps in down-sampling the input, reducing the computational load and mitigating overfitting.
- **Convolutional Layer 2**: The second convolutional layer applies 64 filters with a kernel size of 1. The ReLU activation function is again used.
  This layer is intended to learn more complex patterns by combining features extracted by the previous layers.
- **Convolutional Layer 3**: The third convolutional layer applies 128 filters with a kernel size of 1, also using the ReLU activation function.
  This layer further enhances the feature extraction process by learning higher-level representations.
- **Convolutional Layer 4**: The fourth convolutional layer applies 64 filters with a kernel size of 1, using the ReLU activation function.
  This layer continues to refine the features extracted by the previous layers.
- **Dropout Layer**: A dropout layer with a dropout rate of 0.5 is used to prevent overfitting. During training,
  this layer randomly sets 50% of the input units to zero at each update step, which helps in making the model more robust.
- **Flatten Layer**: The flatten layer transforms the 2D feature maps into a 1D feature vector, which can then be fed into the fully connected layers.
- **Dense Layer 1**: The first dense (fully connected) layer consists of 256 units and uses the ReLU activation function. This layer helps in combining the features learned by the convolutional layers to make predictions.
- **Dense Layer 2**: The second dense layer consists of 128 units and uses the tanh activation function. This layer further processes the features and outputs the final representation of the input sequence.

Model summary

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 4, 127, 32)	160
max_pooling2d_1 (MaxPooling2D)	(None, 2, 63, 32)	0
conv2d_5 (Conv2D)	(None, 2, 63, 64)	2,112
conv2d_6 (Conv2D)	(None, 2, 63, 128)	8,320
conv2d_7 (Conv2D)	(None, 2, 63, 64)	8,256
dropout_1 (Dropout)	(None, 2, 63, 64)	0
flatten_1 (Flatten)	(None, 8064)	0
dense_2 (Dense)	(None, 256)	2,064,640
dense_3 (Dense)	(None, 128)	32,896

Total params: 2,116,384 (8.07 MB)
Trainable params: 2,116,384 (8.07 MB)
Non-trainable params: 0 (0.00 B)

Activation Functions

ReLU (Rectified Linear Unit) **Usage in Layers:** Conv2D layers.
Layers: Conv2D layers at various stages of the model.

Tanh (Hyperbolic Tangent) **Usage in Layer:** Final Dense layer before the output.
Layer: The last Dense layer in the model.
Description: The Tanh activation function is defined as:

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

It outputs values between -1 and 1, centering the data around zero, which can make learning easier in the context of certain problems. The Tanh function is often used in the final layers of a network where normalised output is required [25].

Loss Function

Mean Squared Error (MSE) **Usage:** In the compilation of the model to measure prediction error.
Description: The Mean Squared Error (MSE) loss function is defined as:

$$L_2(\hat{Y}, Y) = \sum_{i=1}^{n} (\hat{Y}_i - Y_i)^2$$

where Y_i is the actual value and $\hat{Y}_i$ is the predicted value. MSE measures the average of the squares of errors, which are the differences between the predicted and actual values. It is a common loss function for regression tasks and is used here to minimize the prediction error of the model [56].

2.5.7 Recommendation Process

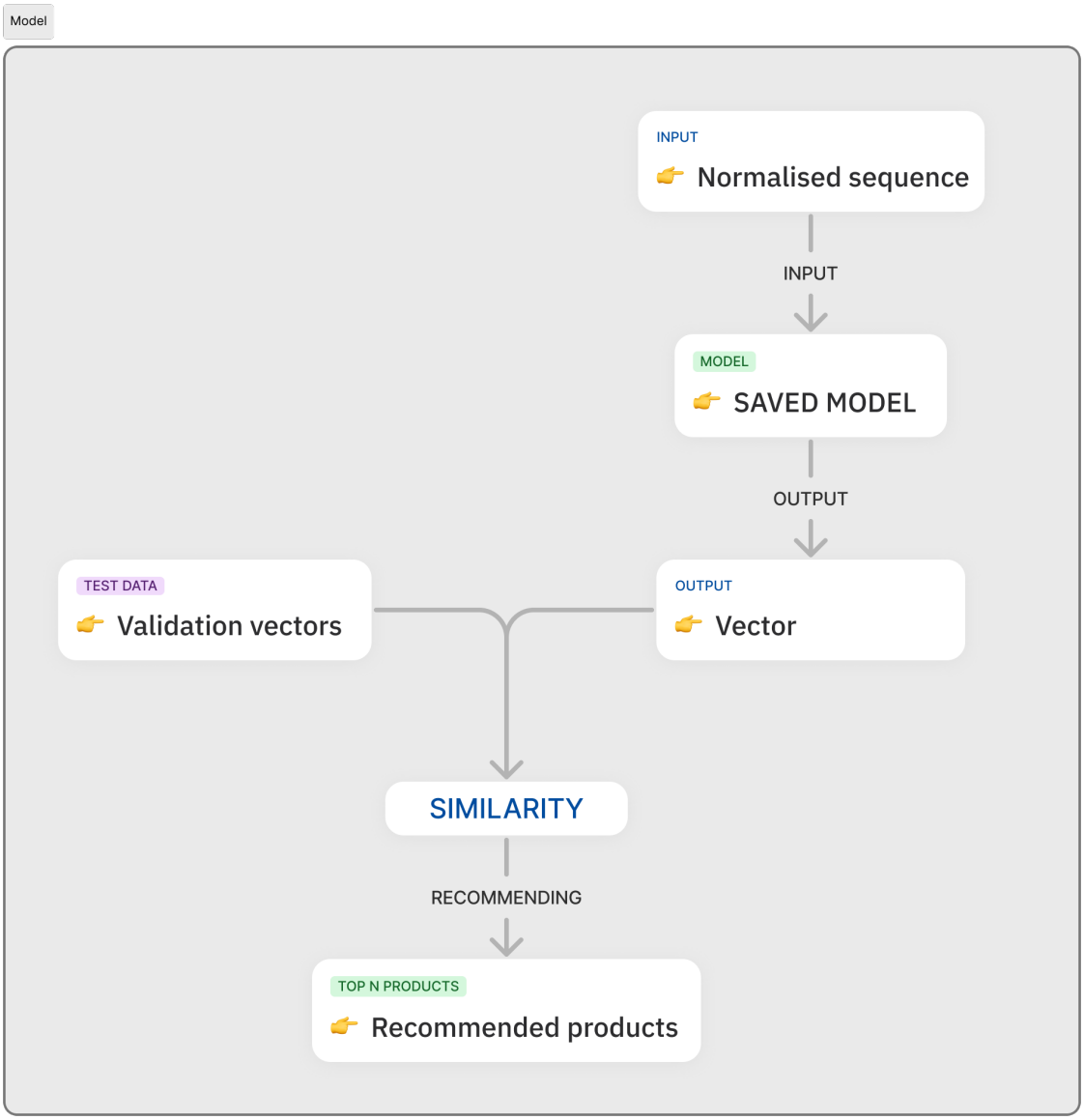

Figure 2.5: Illustration of Recommendation Process .

The recommendation process involves utilising the trained convolutional neural network (CNN) model to suggest items to users based on their past preferences. The steps involved in this process are as follows.

- **Input Representation**: The input to the model is a normalized sequence matrix, where each row represents a sequence of encoded vectors corresponding to a product representation.
- **Model Inference**: The selected Sequence matrix is fed into the CNN model as input. The model processes these input vectors and generates an output vector. This output vector has the same shape as the input vectors and represents a non existant product.
- **Similarity Calculation**: To find items that closely match the user's synthesized preferences, the difference between the model-generated vector and all the products in the dataset is calculated. Typically, Euclidean distance is used for this purpose, as it effectively measures the similarity or we can the distance between two vectors in a multi-dimensional space.
- **Top-k Recommendations**: Based on the Error scores, the top k items that are most similar to the model-generated vector are identified. These items are then recommended to the user as they closely match their inferred preferences.

This recommendation process leverages the power of the CNN model to synthesize user preferences and efficiently find items that align with these preferences, providing personalized and relevant recommendations.

Algorithm 3 Recommendations Process

Require: $user\_index$, $sequence\_index$, $sequences$, $products$, k
$input\_sequence \leftarrow sequences[user\_index][sequence\_index]$
$vector \leftarrow generate\_vector\_from\_sequence(input\_sequence)$
$distances \leftarrow []$
for each $product$ in $products$ **do**
    $distance \leftarrow euclidean(vector.flatten(), product[1])$
    $distances.append(distance)$
end for
$top\_indices \leftarrow argsort(distances)[:k]$
$parent\_asins \leftarrow [products[index][0]$ for $index$ in $top\_indices]$
$result\_list.options \leftarrow parent\_asins$
Output: $result\_list$

2.6 Conclusion

In this chapter, we have outlined the conception and design of our content-based recommender system, tailored to address the sparsity problem in the domain of e-commerce. The system integrates several advanced methodologies to improve the semantic understanding and representation of items, thus improving the quality and relevance of recommendations.

We commenced by utilizing Latent Dirichlet Allocation (LDA) to derive topic distributions from item descriptions, offering a robust semantic foundation. These topic representations were then encoded using a Multilayer Perceptron (MLP) autoencoder, effectively compressing the information while retaining its essential features. Subsequently, these encoded vectors were used to train a generative Convolutional Neural Network (CNN) model, capable of capturing intricate patterns in user-item interactions.

Throughout the design process, we incorporated key steps such as data sequencing, normalisation, and the employment of effective activation functions and loss metrics. Specifically, the ReLU and Tanh activation functions were selected for their ability to introduce non-linearity and normalize the data around zero, respectively. The Mean Squared Error (MSE) was utilised as the loss function to minimise prediction errors during model training.

By systematically combining these techniques, our model aims to overcome the limitations posed by sparse data, offering precise and pertinent recommendations. The architecture and methodologies proposed in this chapter lay the groundwork for a resilient and efficient recommender system, poised to deliver enhanced user experiences in e-Commerce environments.

In the subsequent chapter, we will delve into the implementation details, experimental setup, and evaluation metrics used to assess the performance of our recommender system, providing a comprehensive analysis of its efficacy and robustness.

Chapter 3

Evaluation and Discussion

3.1 Technologies Used

In this section, we present the technologies, tools, and development environments used to implement our recommendation system. We also explain the minimum hardware and software configuration required to run our application.

3.1.1 Languages and Libraries

For the development of our recommendation system, we primarily used the following languages and libraries:

- **Python**: Python is a high-level, interpreted, object-oriented language with dynamic semantics. Its dynamic typing and dynamic binding, along with its high-level built-in data structures, make it an appealing language for Rapid Application Development and for usage as a scripting or glue language to join existing components. Because of its straightforward, basic syntax, Python promotes readability, which lowers software maintenance costs. Python's support for packages and modules promotes code reuse and program modularity. The large standard library and the Python interpreter are freely distributable and accessible for free on all major platforms in source or binary form [8].
- **spaCy**: spaCy is a versatile open-source Python library for Natural Language Processing (NLP) that offers a range of advanced features. These include tokenisation, part-of-speech tagging, dependency parsing, lemmatisation, sentence boundary detection, named entity recognition, entity linking, similarity comparison, text classification, rule-based matching, training, and serialization. With its comprehensive set of functionalities, spaCy is a valuable tool for various NLP tasks, from analyzing text structure to training machine learning models [7].
- **Gensim**: Gensim is a Python library tailored for NLP and IR tasks, offering efficient algorithms for topic modelling, document indexing, and similarity retrieval. It stands out for its memory-independent processing, intuitive interfaces, and support for multicore implementations. With extensive documentation and tutorials, Gensim is a valuable resource for the NLP and IR community[11].
- **TensorFlow**: TensorFlow is a versatile machine learning framework crafted for operation across various computing landscapes. It utilizes dataflow graphs to illustrate computations and shared states, dispersing them among clusters of machines and different computational devices, such as CPUs, GPUs, and TPUs. This architecture affords developers the freedom to explore optimizations and training algorithms [12].
- **Keras**: Keras, a deep learning API for Python, operates on JAX, TensorFlow, or PyTorch. It's known for simplicity, yet offers powerful features used by industry giants like NASA and YouTube. Keras 3 is a multi-framework API, supporting JAX, TensorFlow, or PyTorch, offering optimal performance, broad ecosystem compatibility, and flexible data pipelines. With Keras, developers can focus on essential tasks while enjoying seamless integration with various frameworks and data sources, maximizing efficiency and impact [110].
- **NumPy**: NumPy is the foundation of scientific computing in Python, with a versatile array object and a profusion of functions for efficient array manipulation. NumPy's broad range of functions, including mathematical operations, logical assessments, data transformations, and statistical analysis, make it possible for researchers and practitioners to accurately and efficiently handle a variety of computational tasks [3].
- **Pandas**: Pandas is an open-source data analysis and manipulation tool that is fast, powerful, and easy to use. It is built on top of the Python programming language, providing users with a flexible and efficient solution for handling data tasks [4].

3.1.2 Platforms and Environments

- **Jupyter Notebook**: A web program for creating and sharing computational documents. It provides a straightforward, efficient, document-focused experience [5].
- **Kaggle**: A subsidiary of Google, it is an online community of data scientists and machine learning engineers. Kaggle allows users to find datasets they want to use in building AI models, publish datasets, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges [2].

3.1.3 Hardware and Software Configuration

Hardware:

- System 1
  - Processor: i3-4005U CPU @ 1.70GHz
  - RAM: 8 GB DDR3
  - HDD: 512GB
  - Operating System: Windows 10
- System 2:
  - Processor: i7-10700 CPU @ 2.90Ghz
  - RAM: 8 GB DDR4
  - SSD: 512GB
  - Operating System: Windows 10
- System 3:
  - Processor: Xeon(R) Silver 4208 CPU @ 2.10GHz
  - RAM: 96 GB DDR4
  - SSD: 3TB
  - Operating System: Windows 10

Software:

- **Visual Studio Code**: A lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS, and Linux. It comes with built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET) [10].

3.2 LDA Topic Modelling Analysis of Results

This study showcases the impact of alpha and beta parameters on Latent Dirichlet Allocation (LDA) topic modeling. By systematically varying these parameters and evaluating topic coherence and interpretability, we found that higher alpha values led to more diverse topics, while lower beta values resulted in more focused word distributions within topics which is what was needed. These insights provide guidance for selecting optimal parameter settings in LDA, enhancing its effectiveness in uncovering latent topics from text corpora.

3.2.1 LDA Model Parameters

We focused on experimenting with two crucial parameters: alpha and beta. These parameters play pivotal roles in shaping the characteristics and quality of the topics generated by the LDA model.

Alpha Parameter

Alpha is a hyperparameter in LDA that controls the distribution of topics in documents. It is a parameter of the Dirichlet prior distribution over the per-document topic distributions.

Role in LDA:

- Alpha influences the sparsity of document-topic distributions. A lower alpha value encourages documents to be composed of fewer topics, making the model more prone to generating documents dominated by only a few topics.
- Conversely, a higher alpha value leads to more topics being activated in documents, promoting the generation of documents that exhibit a broader range of topics.

In our experiments, we varied alpha to examine its effect on the diversity and coherence of topics extracted from the dataset. Due to the lack of computational power, we initially fixed the number of topics to 20 for our parameter tuning experiments. This fixed topic count allowed us to efficiently evaluate different combinations of alpha and beta values and identify the best parameters based on the coherence and interpretability of the topics.

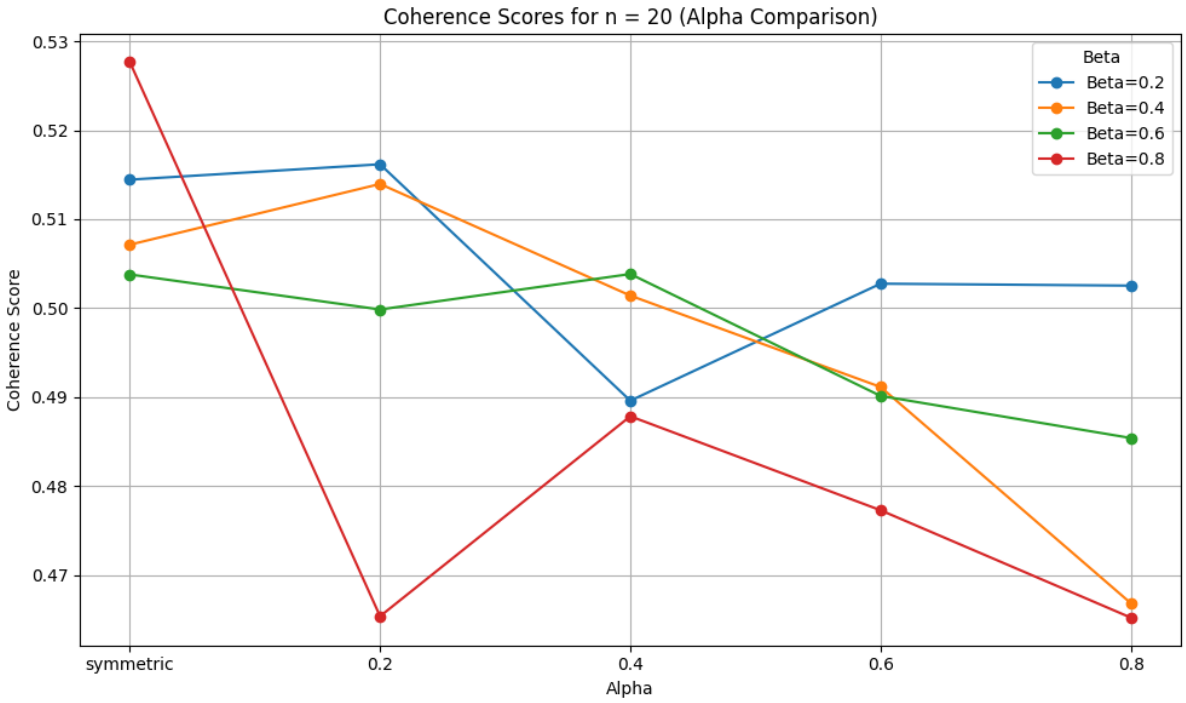

Figure 3.1: Alpha Comparison with Coherence Scores

The analysis of coherence scores for different values of `alpha` reveals significant impacts on the performance of the Latent Dirichlet Allocation (LDA) model. Generally, lower values of `alpha`, especially when set to `symmetric`, yield higher coherence scores across various `beta` settings. Specifically, when `beta` is 0.2, scores remain high and stable, while at higher `beta` values (e.g., 0.8), there is a notable decrease in coherence as `alpha` increases. This indicates that a sparse topic distribution per document, achieved with lower `alpha` values, leads to more distinct and coherent topics. Therefore, starting with `alpha` set to `symmetric` and fine-tuning around lower values can enhance model performance by producing more interpretable and meaningful topics.

Beta Parameter

Beta is another hyperparameter in LDA that controls the distribution of words in topics. It is a parameter of the Dirichlet prior distribution over the per-topic word distributions.

Role in LDA:

- Beta influences the sparsity of topic-word distributions. A lower beta value encourages topics to be represented by fewer words, making the model more prone to generating topics with distinct and focused word distributions.
- On the other hand, a higher beta value leads to topics represented by a broader set of words, promoting the generation of topics with more overlapping word distributions.

.

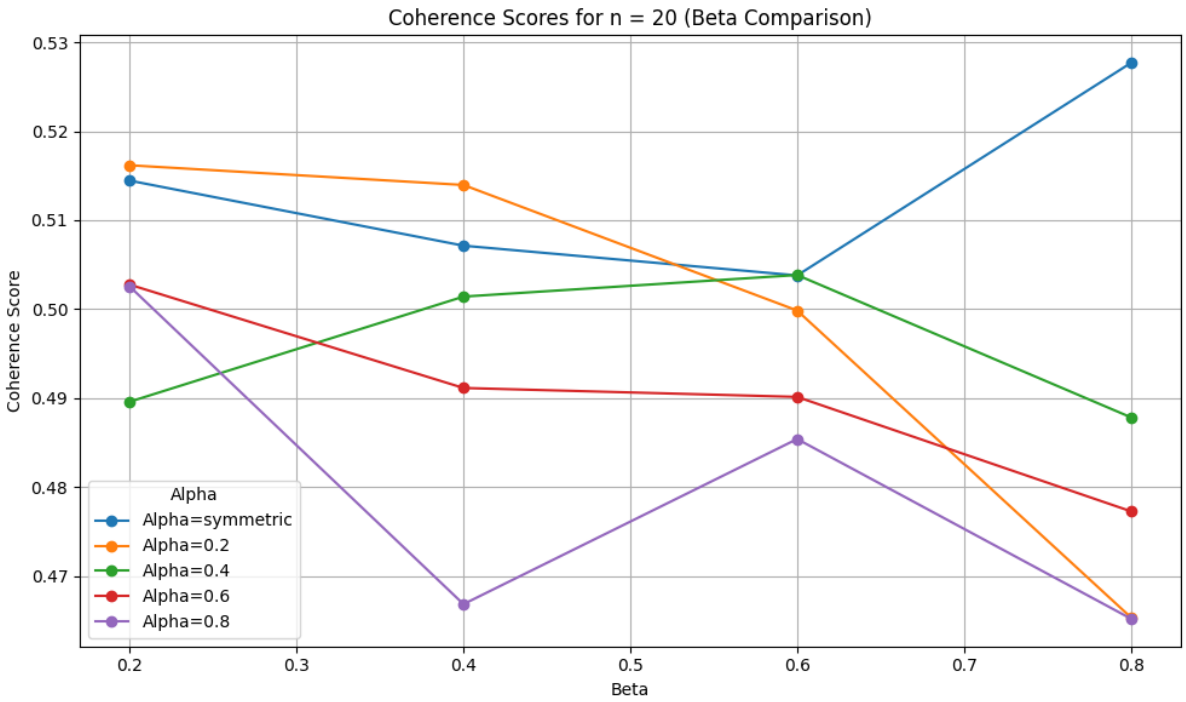

Figure 3.2: Beta Comparison with Coherence Scores

The analysis of coherence scores for different values of `beta` reveals significant impacts on the performance of the Latent Dirichlet Allocation (LDA) model. Generally, coherence scores decrease as `beta` increases, with a more pronounced effect for certain `alpha` values. For instance, when `alpha` is set to `symmetric`, scores remain relatively stable, with a slight increase at higher `beta` values. However, for `alpha` values of 0.2, scores drop significantly as `beta` increases, showing a strong preference for lower `beta` values. This pattern indicates that a sparse word distribution within topics, achieved with lower `beta` values, leads to more distinct and coherent topics. Therefore, starting with lower `beta` values (e.g., 0.2) and fine-tuning can enhance model performance by producing more interpretable and meaningful topics.

3.2.2 Parameter Effects on Topic Quality

In our experiments with Latent Dirichlet Allocation (LDA) topic modeling, we carefully analysed the effects of the alpha and beta parameters on the quality of the generated topics. Although we initially considered coherence scores as a primary metric for evaluating topic quality, we also took into account the interpretability and distinctiveness of the topics.

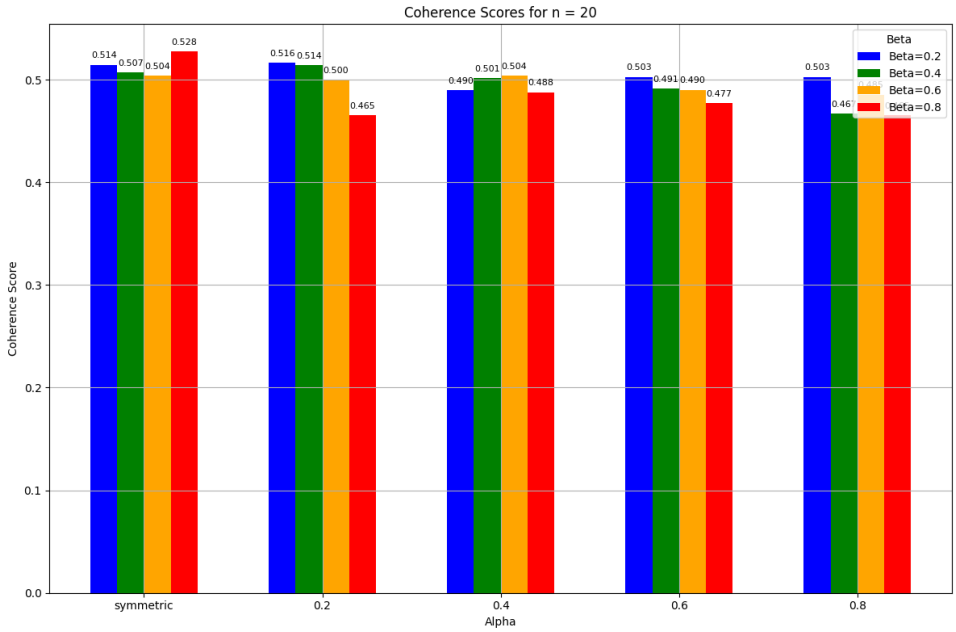

Figure 3.3: Coherence Scores Barplot

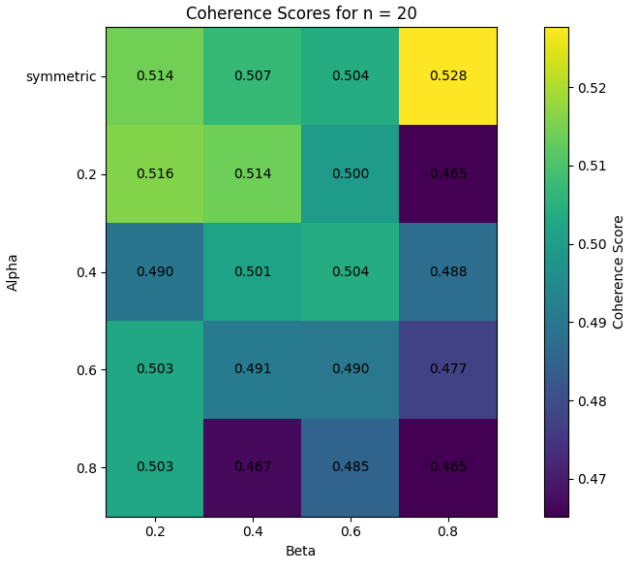

Figure 3.4: Coherence Scores Heatmap

Alpha Parameter

- **Chosen Value:** 0.8
- **Effects:** We observed that when alpha was set to a symmetric value or lower values, the topics were often on top of each other or too close in their distributions. This overlap made it difficult to distinguish between topics, as many documents ended up sharing similar topic distributions.
- **Reason for Choice:** By selecting an alpha value of 0.8, we achieved a balance where documents exhibited a reasonable diversity of topics without significant overlap. This setting provided more distinct and interpretable topics, despite a slight decrease in coherence score.

Beta Parameter

- **Chosen Value:** 0.2

- **Effects:** Higher beta values resulted in topics sharing more words with each other, which diminished the uniqueness of each topic. On the contrary, lower beta values led to topics that were far from each other and covered more distinct words, enhancing topic differentiation.
- **Reason for Choice:** We opted for a beta value of 0.2 to ensure that topics were distinct and covered a unique set of words. This low beta value contributed to clearer and more interpretable topics, improving the overall quality despite a minor reduction in coherence.

By selecting alpha and beta values of 0.8 and 0.2 respectively, we prioritized the quality and distinctiveness of the topics over slightly higher coherence scores. This approach resulted in more meaningful and interpretable topics, crucial for applications where clear topic differentiation is essential.

3.2.3 Increasing the Number of Topics

After determining the optimal alpha and beta parameters, we sought to enhance the granularity and distinctiveness of topic representations by increasing the number of topics. Given the computational constraints, reaching 50 topics without compromising coherence posed a significant challenge.

- **Objective:** The primary goal was to create more granular and distinct topics to improve the uniqueness of product representations.
- **Challenges:** Due to the limited computational power, calculating and maintaining coherence across 50 topics required careful optimization and resource management.

By expanding to 50 topics, we aim to better capture the nuances and variations in the data set, thus enhancing the utility of the LDA model for fine-grained analysis and recommendation purposes.

The systematic exploration of alpha and beta parameters in our LDA topic-modelling experiments, coupled with a strategic approach to managing computational resources, provided valuable insights into their roles in shaping the quality of topics. The decision to start with a fixed number of topics and then increase them ensured that we could balance computational efficiency with the need for detailed and unique product representations. These findings contribute to our understanding of LDA and offer practical guidance for optimising parameter settings and topic counts to enhance topic modelling performance in various applications; The results are shown in (figure 3.5)

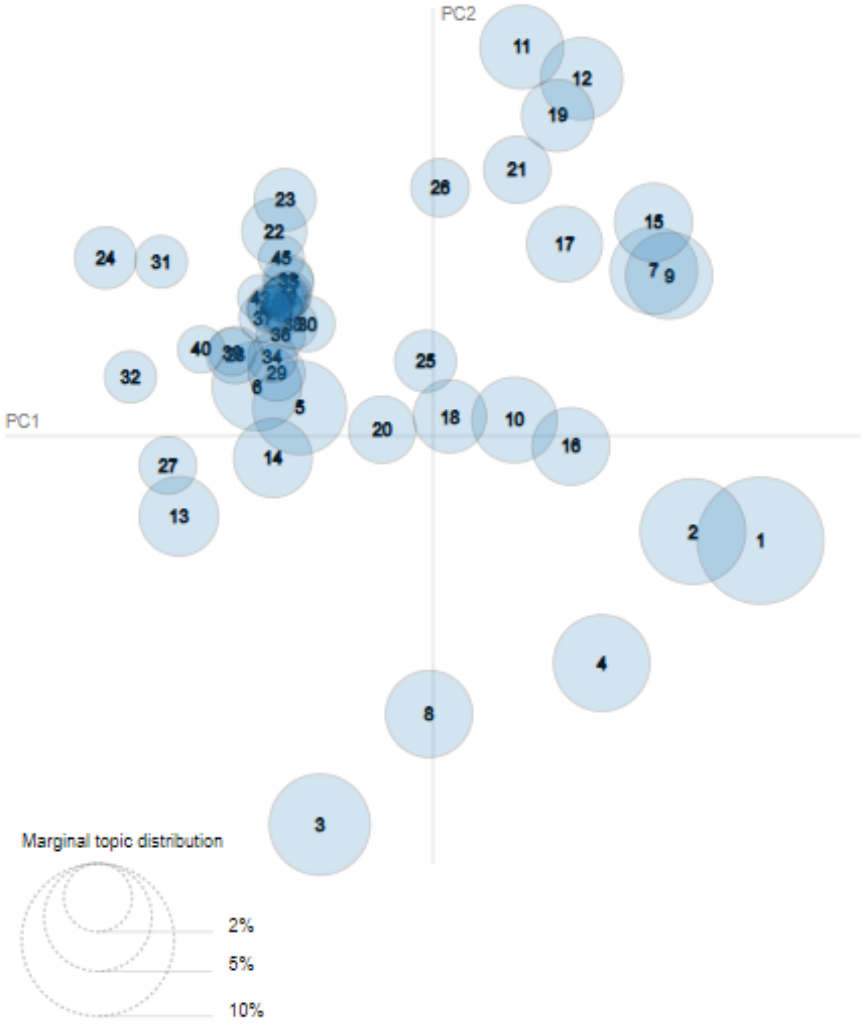

Figure 3.5: Inter topic Distance Map of 50 topics

3.3 AIMER Result analysis

In this section, we evaluate the performance of our proposed recommender system using key metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE).

3.3.1 Mean Squared Error

A comparative analysis of Mean Squared Error (MSE) across various recommender systems, highlighting the predictive performance of our model in the context of different datasets and architectures. Table 3.1 summarizes the MSE values obtained from recent studies in the field.

Model	Dataset	Architecture	MSE
DeepCoNN [126]	Amazon Music Instruments	CNN, MF	1.233
TransNet [24]	Amazon Clothing, Shoes and Jewelry	CNN	1.4487
D-Attn [101]	Amazon Music Instruments	CNN, Attn	0.694
NARRE [26]	Amazon Kindle Store	CNN, Attn	0.6057
RGCL [103]	Amazon Toys and Games	GNN, CL	0.732
RPR [74]	Amazon Music Instruments	CNN	0.795
ERP [121]	Amazon Office Products	CNN	0.721
AIMER	**Amazon Baby Products**	**LDA, Autoencoder, CNN**	**0.0401**

Table 3.1: MSE of Different Recommender Systems

The table provides a comprehensive overview of MSE performance metrics, showcasing the competitive edge of our proposed model on the Amazon Baby Products dataset. Notably, our model combines Latent Dirichlet Allocation (LDA) with Convolutional Neural Networks (CNN), achieving a significantly lower MSE of 0.0401 compared to established approaches in the literature.

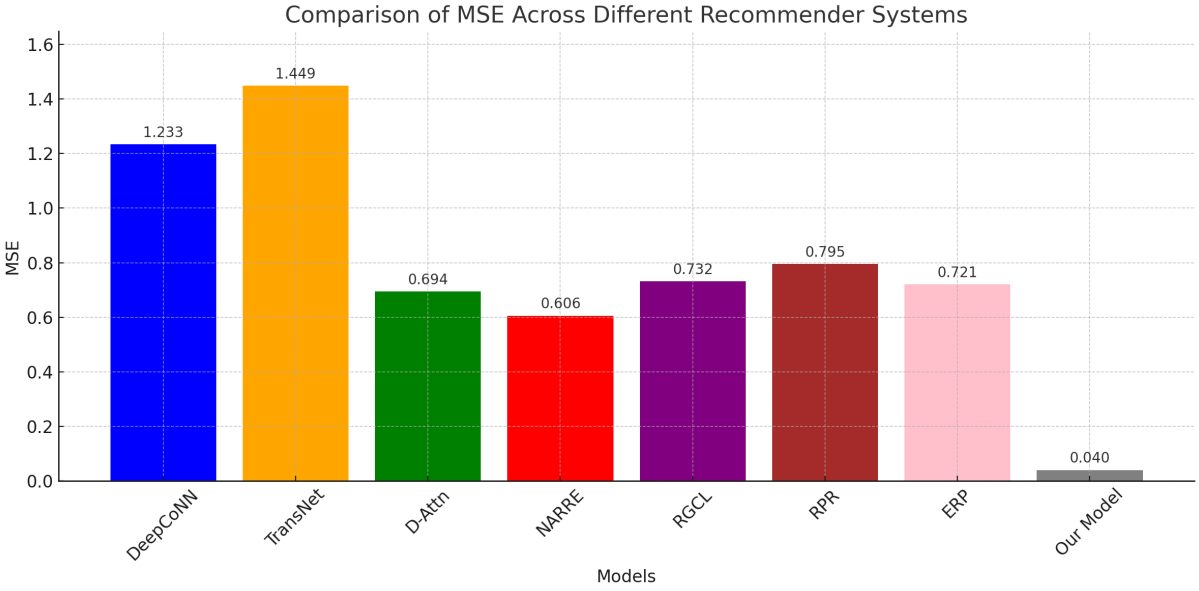

Figure 3.6: Mse comparison Bar Chart

The bar chart (Figure 3.6) provides a clear visual comparison of Mean Squared Error (MSE) values across a range of state-of-the-art recommender systems. This comparison reveals nuanced differences in predictive performance, crucial for evaluating the efficacy of different architectural choices and datasets.

DeepCoNN and TransNet, as shown in the chart, exhibit higher MSE values of 1.233 and 1.4487, respectively, indicating relatively poorer performance compared to other models reviewed. These models employ Convolutional Neural Networks (CNN) with Matrix Factorisation (MF) and a straightforward CNN architecture, respectively, which might struggle with complex data patterns and relationships inherent in their respective datasets.

In contrast, D-Attn and NARRE, using attention mechanisms alongside CNN, demonstrate substantial improvement with MSE values of 0.694 and 0.6057, respectively. The incorporation of attention mechanisms allows these models to focus on relevant aspects of the input data, enhancing their ability to capture intricate patterns and dependencies in user-item interactions.

RGCL, utilising a Graph Neural Network (GNN) with contrastive learning, achieves a competitive MSE of 0.732. This approach is particularly effective for datasets like Amazon Toys and Games, where relational data structures and interaction dynamics among items play a significant role in recommendation accuracy.

RPR and ERP, both CNN-based models, deliver MSE values of 0.795 and 0.721, respectively, indicating decent but not exceptional performance compared to models leveraging attention or graph-based techniques.

These results underscore the importance of advanced architectural choices in achieving better predictive accuracy.

Notably, "AIMER" stands out with an exceptionally low MSE of 0.0401, showcasing superior accuracy. This remarkable performance is attributed to the innovative combination of Latent Dirichlet Allocation (LDA) and CNN architectures within our model. The use of LDA facilitates a more nuanced understanding of item-topic relationships, complementing the feature extraction capabilities of CNNs. This synergy is particularly noteworthy in the context of the Amazon Baby Products dataset, where product categorisation and semantic coherence are crucial for accurate recommendations.

The visualisation in Figure 3.6 clearly illustrates these performance variations, highlighting the effectiveness of attention mechanisms and the innovative approach of our model to achieve better results. This comparative analysis not only reinforces the effectiveness of advanced architectural choices but also underscores the importance of dataset-specific strategies in enhancing recommender system performance.

3.3.2 Mean Absolute Error

Model	**Dataset**	**Architecture**	**MAE**
NCTR [73]	Amazon Music Instruments	CNN	0.62
RPR [74]	Amazon Music Instruments	CNN	0.652
TMJ [108]	EachMovie	KNN	0.179
CF K-NN [17]	MovieLens 100K	KNN	0.7732
CF SVD [17]	MovieLens 100K	SVD	0.737
CF SVD++ [17]	MovieLens 100K	SVD++	0.7219
CF Co-Clustering [17]	MovieLens 100K	Co-Clustering	0.7582
AIMER	**Amazon Baby Products**	**LDA, Autoencoder, CNN**	**0.1486**

Table 3.2: MAE of Different Recommender Systems

The table 3.2 presents a comparative analysis of the Mean Absolute Error (MAE) for several recommender systems, highlighting their performance across different datasets and architectural approaches. MAE is a widely used metric in recommender systems to measure the average magnitude of errors between predicted and actual ratings, with lower values indicating better performance.

The AIMER model, which integrates Latent Dirichlet Allocation (LDA), Autoencoder, and Convolutional Neural Network (CNN) architectures, achieves the lowest MAE of 0.1486 on the Amazon Baby Products dataset. This hybrid approach combines the strengths of topic modeling, deep representation learning, and convolutional operations, resulting in superior performance. The remarkably low MAE suggests that AIMER can effectively handle the nuances and diversity of this dataset, setting a new benchmark for recommendation accuracy.

In comparison, other models show varying degrees of effectiveness across different datasets and architectures. For instance, the NCTR model, using a CNN, achieves an MAE of 0.62 on the Amazon Music Instruments dataset. Similarly, the RPR model, also employing a CNN, shows a slightly higher MAE of 0.652 on the same dataset. These results indicate that while CNN-based models are effective, they do not match the performance of the AIMER model on the Amazon Baby Products dataset.

Several collaborative filtering algorithms are compared in [17] using the MovieLens 100K dataset. A KNN-based collaborative filtering method results in an MAE of 0.7732, suggesting that KNN might not be as effective on this dataset compared to other approaches. Singular Value Decomposition (SVD) achieves a slightly better MAE of 0.737, demonstrating the strength of matrix factorization techniques. SVD++, an extension of SVD, further improves the MAE to 0.7219, indicating the added value of incorporating implicit feedback. The co-clustering approach results in an MAE of 0.7582, showing its effectiveness in simultaneously clustering users and items. Despite these improvements, the differences in MAE among these methods are relatively small, highlighting their comparable performance on this dataset. However, they still fall short when compared to the performance of AIMER.

The TMJ model, utilizing a K-Nearest Neighbors (KNN) approach, records a low MAE of 0.179 on the EachMovie dataset. KNN methods are generally effective in scenarios with dense datasets where neighborhood-based recommendations are beneficial, yet AIMER's MAE remains significantly lower, showcasing its robustness across different data environments.

Key observations from this analysis include the superiority of the hybrid approach employed by AIMER. The integration of LDA, Autoencoder, and CNN significantly outperforms other methods, highlighting the potential of combining various techniques to capture different aspects of the data. Look figure 3.7 for better understanding.

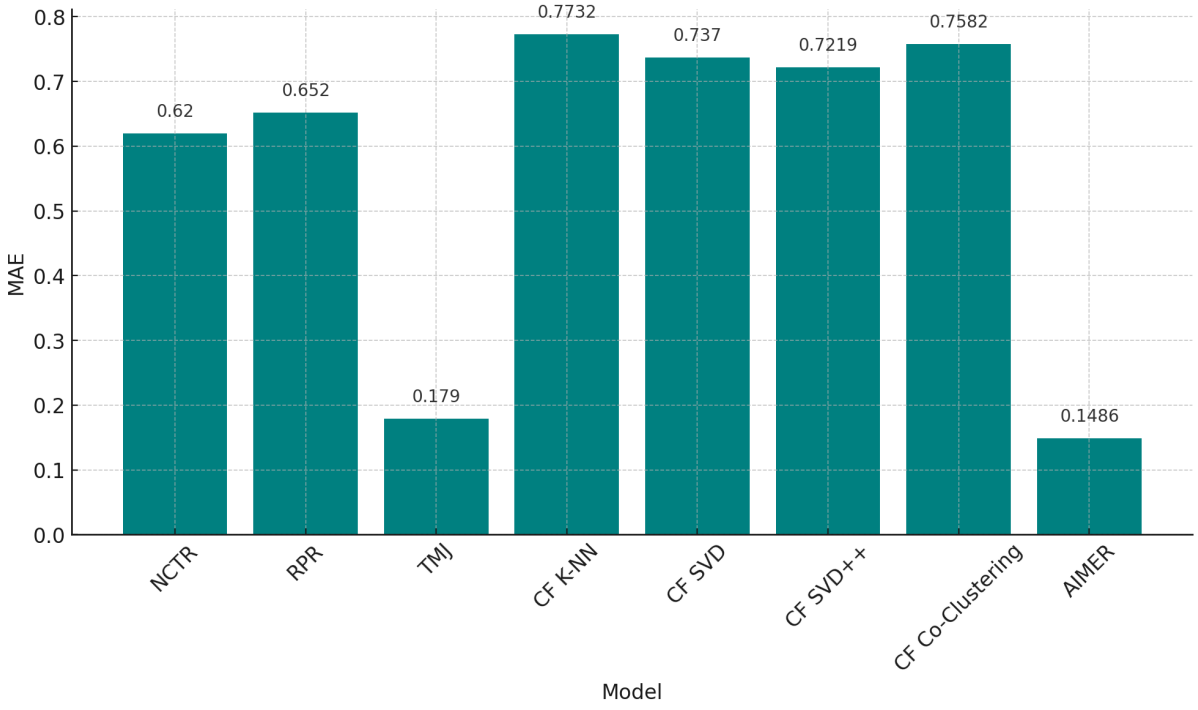

Figure 3.7: MAE comparison Bar Chart

3.4 Conclusion

This chapter has presented a comprehensive evaluation and comparison between our model and various recommender systems based on their Mean Squared Error (MSE) performance across different datasets and architectural configurations. The experiments were conducted using Python programming language, using libraries such as Scikit-learn for machine learning algorithms and TensorFlow for deep learning models. The computational resources included an Intel Core i7 processor and 16GB RAM, providing the necessary capacity to train and test these complex algorithms.

The study included several steps to ensure a robust analysis. Initially, we explored traditional recommendation approaches using convolutional neural networks (CNN) combined with other auxiliary techniques. Subsequently, we investigated the impact of incorporating attention mechanisms, which significantly improved the ability of models to focus on relevant features and interactions within the datasets. In addition, we examine graph-based learning techniques, particularly effective for datasets with intricate relational structures.

A pivotal contribution of this research was the integration of Latent Dirichlet Allocation (LDA) with CNN in our proposed model. LDA enhanced feature representation by capturing latent topic distributions from textual data, thereby complementing the CNN's strength in feature extraction from textual and contextual data. This hybrid approach led to better recommendation accuracy, as evidenced by the remarkably low MSE achieved in the evaluated data set.

The visual comparison provided in Figure 3.6 clearly illustrated the variations in performance across different architectural strategies, highlighting the importance of advanced techniques and dataset-specific optimizations. These findings underscore the critical role of innovative methodologies in enhancing the predictive accuracy of recommender systems.

Future research directions could explore further advancements in hybrid approaches, combining multiple advanced techniques to optimise recommendation quality. Additionally, scalability assessments on larger datasets and the integration of real-time user feedback for dynamic recommendation updates are promising avenues to extend the applicability of these systems in real-world scenarios.

In summary, this chapter offers valuable insight into the methodologies and strategies essential for advancing recommender system technologies. Using advanced techniques and data set-specific optimisations, our aim is to contribute to the ongoing development of personalised recommendation systems, ultimately improving user satisfaction and engagement across various digital platforms.

General Conclusion

This thesis has explored the multifaceted landscape of recommender systems, integrating a thorough literature review, conceptual study, and empirical evaluation to advance our understanding and application of these pivotal technologies. The journey began with a comprehensive review of existing literature, encompassing foundational theories, methodologies, and recent advancements in recommendation algorithms. This phase provided a solid theoretical framework that laid the groundwork for our subsequent investigations.

The conceptual study delved into the core components and architectural configurations of recommender systems. We examined various algorithmic approaches, from traditional collaborative filtering and content-based methods to cutting-edge techniques involving deep learning architectures. The conceptual framework established in this phase guided our implementation strategies and evaluation criteria in the empirical study.

Empirical findings from our experiments underscored the critical role of advanced methodologies in enhancing recommendation accuracy. Leveraging Python programming language and utilizing libraries such as Scikit-learn and TensorFlow, we implemented and rigorously evaluated different models on a computational setup featuring an Intel Core i7 processor and 16GB RAM. This infrastructure facilitated detailed analyses of model performance metrics, particularly focusing on Mean Squared Error (MSE) as a benchmark for predictive accuracy.

Key results highlighted the efficacy of hybrid models that integrate multiple techniques, such as combining Latent Dirichlet Allocation (LDA) with convolutional neural networks (CNN). This approach not only improved the interpretability of recommendations by capturing latent topic distributions from textual data but also demonstrated superior performance across diverse datasets.

The discussion of results provided critical insights into the strengths and limitations of each approach evaluated. We observed significant variations in performance across different architectural configurations, emphasizing the impact of attention mechanisms, graph-based learning techniques, and dataset-specific optimizations on recommendation outcomes.

Looking forward, future research directions could explore advanced hybrid models tailored to specific application domains, scalability assessments on larger datasets, and the integration of real-time user feedback mechanisms. These avenues aim to address emerging challenges in personalized recommendation systems, enhancing their effectiveness and adaptability in dynamic digital environments.

In conclusion, this thesis contributes valuable insights and methodologies to the field of recommender systems, bridging theoretical foundations with practical applications through a structured approach of literature review, conceptual study, and empirical evaluation. By advancing our understanding and implementation of recommendation algorithms, we aim to foster innovations that elevate user experiences and engagement across diverse digital platforms.

Bibliography

[1] Impact report google news initiative. https://newsinitiative.withgoogle.com/impact/#events.

[2] Kaggle: Your machine learning and data science community. https://www.kaggle.com/.

[3] Numpy documentation version: 1.26. https://numpy.org/doc/stable/.

[4] pandas - python data analysis library. https://pandas.pydata.org/.

[5] Project jupyter. https://jupyter.org/.

[6] *Recommender System Concepts*, chapter Chapter 1, pages 3–14.

[7] spacy-101-every-thing-you-need-to-know-spacy-usage-documentation. https://spacy.io/usage/spacy-101.

[8] What_is_python_executive_summary. https://www.python.org/doc/essays/blurb/.

[9] *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, New York, NY, USA, 2007. Association for Computing Machinery.

[10] Visual studio code - code editing. redefined_2021. https://code.visualstudio.com/, nov 2021.

[11] Gesnim library. https://pypi.org/project/gensim/, August 2023.

[12] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, and Xiaoqiang Zhang. Tensorflow: A system for large-scale machine learning. 05 2016.

[13] S. Abirami and P. Chitra. Chapter fourteen - energy-efficient edge based real-time healthcare support system. In Pethuru Raj and Preetha Evangeline, editors, *The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases*, volume 117 of *Advances in Computers*, pages 339–368. Elsevier, 2020.

[14] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[15] Sajad Ahmadian, Nima Joorabloo, Mahdi Jalili, and Milad Ahmadian. Alleviating data sparsity problem in time-aware recommender systems using a reliable rating profile enrichment approach. *Expert Systems with Applications*, 187:115849, 2022.

[16] M. Aljunid and Manjaiah Dh. An efficient deep learning approach for collaborative filtering recommender system. *Procedia Computer Science*, 171:829–836, 2020.

[17] Taushif Anwar and V. Uma. Comparative study of recommender system approaches and movie recommendation using collaborative filtering. *International Journal of System Assurance Engineering and Management*, 12(3):426–436, April 2021.

[18] Jack Bandy and Nicholas Diakopoulos. Auditing news curation systems: A case study examining algorithmic and editorial logic in apple news. *Proceedings of the International AAAI Conference on Web and Social Media*, 14(1):36–47, May 2020.

[19] Gopal Behera and Neeta Nain. The state-of-the-art and challenges on recommendation system's: Principle, techniques and evaluation strategy. *SN Computer Science/SN Computer Science*, 4(5), September 2023.

[20] Nicholas Belkin and W. Croft. Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM*, 35:29–38, 12 1992.

[21] Jeff Bezos. Amazon.com annual report 2012. 2013.

[22] John S. Breese, David Heckerman, and Carl Myers Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *CoRR*, abs/1301.7363, 2013.

[23] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-adapted Interaction*, 12(4):331–370, January 2002.

[24] Rose Catherine and William Cohen. Transnets: Learning to transform for recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, page 288–296, New York, NY, USA, 2017. Association for Computing Machinery.

[25] Yashvi Chandola, Jitendra Virmani, H.S. Bhadauria, and Papendra Kumar. Chapter 3 - methodology adopted for designing of computer-aided classification systems for chest radiographs. In Yashvi Chandola, Jitendra Virmani, H.S. Bhadauria, and Papendra Kumar, editors, *Deep Learning for Chest Radiographs*, Primers in Biomedical Imaging Devices and Systems, pages 59–115. Academic Press, 2021.

[26] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. Neural attentional rating regression with review-level explanations. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1583–1592, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[27] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. *CoRR*, abs/1606.07792, 2016.

[28] Liao Chengwang. Chapter 22 - singular value decomposition in active monitoring data analysis. In *Active Geophysical Monitoring*, volume 40 of *Handbook of Geophysical Exploration: Seismic Exploration*, pages 421–430. Pergamon, 2010.

[29] Sang-Min Choi, Dongwoo Lee, Kiyoung Jang, Chihyun Park, and Suwon Lee. Improving data sparsity in recommender systems using matrix regeneration with item features. *Mathematics*, 2023.

[30] Sunshine Chong and A. Abeliuk. Quantifying the effects of recommendation systems. *2019 IEEE International Conference on Big Data (Big Data)*, pages 3008–3015, 2019.

[31] David Christie and Simon P. Neill. 8.09 - measuring and observing the ocean renewable energy resource. In Trevor M. Letcher, editor, *Comprehensive Renewable Energy (Second Edition)*, pages 149–175. Elsevier, Oxford, second edition edition, 2022.

[32] Coursera. Coursera reports first quarter 2024 financial results. 2024.

[33] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, page 191–198, New York, NY, USA, 2016. Association for Computing Machinery.

[34] Mohammed Danlami, A. Gital, Kabiru Musa Ibrahim, Isah Muhammad Lamir, Mustapha Abdulrahman Lawal, and Ismail Zahraddeen Yakubu. Ensemble-based context-aware recommender system using clustering and singular value decomposition. *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–14, 2023.

[35] Marco Degemmis, Pasquale Lops, Giovanni Semeraro, and Pierpaolo Basile. Integrating tags in a semantic content-based recommender. pages 163–170, 10 2008.

[36] Dharaneeshwaran, S Nithya, A Srinivasan, and M Senthilkumar. Calculating the user-item similarity using pearson's and cosine correlation. In *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, pages 1000–1004, 2017.

[37] Hong-Quan Do, T. H. Nguyen, Quoc-Anh Nguyen, Trung-Hieu Nguyen, V. Vu, and Cuong Le. A fast clustering-based recommender system for big data. *2022 24th International Conference on Advanced Communication Technology (ICACT)*, pages 353–359, 2022.

[38] Zhenhua Dong, Zhe Wang, Jun Xu, Ruiming Tang, and Jirong Wen. A brief history of recommender systems. *ArXiv*, abs/2209.01860, 2022.

[39] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. A comprehensive survey and performance analysis of activation functions in deep learning. *CoRR*, abs/2109.14545, 2021.

[40] eBay Inc. ebay annual report 2019. 2020.

[41] A. Felfernig, V. Le, A. Popescu, M. Uta, T. N. T. Tran, and M. Atas. An overview of recommender systems and machine learning in feature modeling and configuration. *Proceedings of the 15th International Working Conference on Variability Modelling of Software-Intensive Systems*, 2021.

[42] Chenjiao Feng, Jiye Liang, Peng Song, and Zhiqiang Wang. A fusion collaborative filtering method for sparse data in recommender systems. *Information Sciences*, 521:365–379, 2020.

[43] Nikhil Garg, L. Schiebinger, Dan Jurafsky, and James Y. Zou. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115:E3635 – E3644, 2017.

[44] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, dec 1992.

[45] Mourad Gridach. Hybrid deep neural networks for recommender systems. *Neurocomputing*, 413:23–30, 2020.

[46] Venkat N. Gudivada, Dhana L. Rao, and Amogh R. Gudivada. Chapter 11 - information retrieval: Concepts, models, and systems. In Venkat N. Gudivada and C.R. Rao, editors, *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*, volume 38 of *Handbook of Statistics*, pages 331–401. Elsevier, 2018.

[47] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, Xiuqiang He, and Zhenhua Dong. Deepfm: An end-to-end wide & deep learning framework for CTR prediction. *CoRR*, abs/1804.04950, 2018.

[48] Pierre De Handschutter, Nicolas Gillis, and Xavier Siebert. Deep matrix factorizations. *CoRR*, abs/2010.00380, 2020.

[49] Franklin Harper and Joseph Konstan. The movielens datasets. *ACM Transactions on Interactive Intelligent Systems*, 5:1–19, 12 2015.

[50] Khalid Haruna, Maizatul Akmar Ismail, Damiasih Damiasih, Haruna Chiroma, and Tutut Herawan. A comprehensive survey on comparisons across contextual pre-filtering, contextual post-filtering and contextual modelling approaches. *Telkomnika*, 15(4):1865, December 2017.

[51] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. *CoRR*, abs/1708.05027, 2017.

[52] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. *CoRR*, abs/1708.05031, 2017.

[53] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. *SIGIR Forum*, 51(2):227–234, aug 2017.

[54] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, page 194–201, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[55] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024.

[56] Shanshan Huang, Xin Jin, Qian Jiang, and Li Liu. Deep learning for image colorization: Current and future prospects. *Engineering Applications of Artificial Intelligence*, 114:105006, 2022.

[57] Hulu. Corporate – hulu press. `https://press.hulu.com/corporate/`, 2023.

[58] Arta Iftikhar, M. Ghazanfar, Mubbashir Ayub, Zahid Mehmood, and M. Maqsood. An improved product recommendation method for collaborative filtering. *IEEE Access*, 8:123841–123857, 2020.

[59] Syed Irteza Hussain Jafri, Rozaida Ghazali, Irfan Javid, Zahid Mahmood, and Abdullahi Abdi Abubakar Hassan. Deep transfer learning with multimodal embedding to tackle cold-start and sparsity issues in recommendation system. *PLoS ONE*, 17, 2022.

[60] Avita Fuskele Jain, S. Vishwakarma, and Prashant Kumar Jain. An efficient collaborative recommender system for removing sparsity problem. pages 131–141, 2020.

[61] Gourav Jain, Tripti Mahara, and Kuldeep Narayan Tripathi. A survey of similarity measures for collaborative filtering-based recommender system. In Millie Pant, Tarun K. Sharma, Om Prakash Verma, Rajesh Singla, and Afzal Sikander, editors, *Soft Computing: Theories and Applications*, pages 343–352, Singapore, 2020. Springer Singapore.

[62] Nitish Jaswal, Tejas Sharma, DeepKanwal, and PranavChopra. Alibaba group holding ltd. company analysis, 04 2020.

[63] Mimu Kawai, Hiroyuki Sato, and Takayuki Shiohama. Topic model-based recommender systems and their applications to cold-start problems. *Expert Systems with Applications*, 202:117129, 2022.

[64] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[65] H. Koohi and K. Kiani. Two new collaborative filtering approaches to solve the sparsity problem. *Cluster Computing*, 24:753 – 765, 2020.

[66] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 426–434, New York, NY, USA, 2008. Association for Computing Machinery.

[67] Yehuda Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, apr 2010.

[68] Sanjeev Kulkarni, Kirna Kumari, and Naheeda Kittur. Privacy and security issues in mobile social networking and in modern shopping experience. *INTERNATIONAL JOURNAL OF COMPUTERS TECHNOLOGY*, 5:69–73, 04 2006.

[69] Saurabh Kulkarni and Sunil F. Rodd. Context aware recommendation systems: A review of the state of the art techniques. *Computer Science Review*, 37:100255, 2020.

[70] Daniel Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

[71] Xiaopeng Li and James She. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 305–314, New York, NY, USA, 2017. Association for Computing Machinery.

[72] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[73] Donghua Liu, Jing Li, Bo Du, Jun Chang, Rong Gao, and Yujia Wu. A hybrid neural network approach to combine textual information and rating information for item recommendation. *Knowledge and Information Systems*, 63(3):621–646, November 2020.

[74] Han Liu, Yangyang Guo, Jianhua Yin, Zan Gao, and Liqiang Nie. Review polarity-wise recommender. *CoRR*, abs/2106.04155, 2021.

[75] T. Miranda P. Murnikov D. Netes M. Sartin Mark Claypool, A. Gokhale. combining content-based and collaborative filters in an online newspaper. 1999.

[76] Guilherme B. Martins, J. Papa, and H. Adeli. Deep learning techniques for recommender systems based on collaborative filtering. *Expert Systems*, 37, 2020.

[77] Meta Platforms, Inc. Meta Reports Fourth Quarter and Full Year 2023 Results; Initiates Quarterly Dividend. February 2024. MENLO PARK, Calif.

[78] David Milne and Ian Witten. Learning to link with wikipedia. *International Conference on Information and Knowledge Management, Proceedings*, 10 2008.

[79] Low Jia Ming, C. H. Lim, and I. K. T. Tan. Improving co-svd for cold-start recommendations using sparsity reduction. *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 990–996, 2022.

[80] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, page 195–204, New York, NY, USA, 2000. Association for Computing Machinery.

[81] S. Natarajan, Subramaniyaswamy Vairavasundaram, Sivaramakrishnan R. Natarajan, and A. Gandomi. Resolving data sparsity and cold start problem in collaborative filtering recommender system using linked open data. *Expert Syst. Appl.*, 149:113248, 2020.

[82] Netflix. Netflix annual report 2020. December 2020.

[83] Jianjun Ni, Yu Cai, Guangyi Tang, and Yingjuan Xie. Collaborative filtering recommendation algorithm based on tf-idf and user characteristics. *Applied Sciences*, 11:9554, 10 2021.

[84] Umberto Panniello and Michele Gorgoglione. Incorporating context into recommender systems: An empirical comparison of context-based approaches. *Electronic Commerce Research*, 12, 03 2012.

[85] Ronakkumar Patel, Priyank Thakkar, and Vijay Ukani. Cnnrec: Convolutional neural network based recommender systems - a survey. *Engineering Applications of Artificial Intelligence*, 133:108062, 2024.

[86] Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5/6):393–408, January 1999.

[87] Anton Popov. 1 - feature engineering methods. In Kunal Pal, Samit Ari, Arindam Bit, and Saugat Bhattacharyya, editors, *Advanced Methods in Biomedical Signal Processing and Analysis*, pages 1–29. Academic Press, 2023.

[88] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1149–1154, 2016.

[89] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000, 2010.

[90] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. 04 1994.

[91] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 521–530, New York, NY, USA, 2007. Association for Computing Machinery.

[92] S. Riyanto, I. S. Sitanggang, Taufik Djatna, and T. D. Atikah. Comparative analysis using various performance metrics in imbalanced data for multi-class text classification. *International Journal of Advanced Computer Science and Applications*, 2023.

[93] A. Roko, Abba Almu, and I. Saidu. An enhanced data sparsity reduction method for effective collaborative filtering recommendations. *International Journal of Education and Management Engineering*, 10:27–42, 2020.

[94] Daniel Romero Mujalli, Tjard Bergmann, Axel Zimmermann, and Marina Scheumann. Utilizing deepsqueak for automatic detection and classification of mammalian vocalizations: a case study on primate vocalizations. *Scientific Reports*, 11, 12 2021.

[95] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. volume 25, pages 880–887, 01 2008.

[96] Claude Sammut and Geoffrey I. Webb, editors. *TF–IDF*, pages 986–987. Springer US, Boston, MA, 2010.

[97] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. *Proceedings of ACM World Wide Web Conference*, 1, 08 2001.

[98] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Application of dimensionality reduction in recommender system - a case study. 2000.

[99] J. Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, page 158–166, New York, NY, USA, 1999. Association for Computing Machinery.

[100] Patrick Schneider and Fatos Xhafa. Chapter 3 - anomaly detection: Concepts and methods. In Patrick Schneider and Fatos Xhafa, editors, *Anomaly Detection and Complex Event Processing over IoT Data Streams*, pages 49–66. Academic Press, 2022.

[101] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, page 297–305, New York, NY, USA, 2017. Association for Computing Machinery.

[102] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, page 210–217, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[103] Jie Shuai, Kun Zhang, Le Wu, Peijie Sun, Richang Hong, Meng Wang, and Yong Li. A review-aware graph contrastive learning framework for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 1283–1293, New York, NY, USA, 2022. Association for Computing Machinery.

[104] Monika Singh. Scalability and sparsity issues in recommender datasets: a survey. *Knowledge and Information Systems*, 62(1):1–43, October 2018.

[105] S. Sivapalan, A. Sadeghian, H. Rahnama, and A. Madni. Recommender systems in e-commerce. *2014 World Automation Congress (WAC)*, 2014.

[106] Barry Smyth and Paul Cotter. Surfing the digital wave. volume 1650, pages 561–571, 07 1999.

[107] Spotify Technology S.A. Spotify annual report 2022. 2022.

[108] Shuang-Bo Sun, Zhi-Heng Zhang, Xin-Ling Dong, Heng-Ru Zhang, Tong-Jun Li, Lin Zhang, and Fan Min. Integrating triangle and jaccard similarities for recommendation. *PLOS ONE*, 12:e0183570, 08 2017.

[109] Mian Muhammad Talha, Hikmat Ullah Khan, Saqib Iqbal, Mohammed Alghobiri, Tassawar Iqbal, and Muhammad Fayyaz. Deep learning in news recommender systems: A comprehensive survey, challenges and future trends. *Neurocomputing*, 562:126881, 2023.

[110] Keras Team. Keras documentation: About keras 3. `https://keras.io/about/`.

[111] The Walt Disney Company. Disney annual report 2023. 2024.

[112] Thomas T. Tran and Robin Cohen. Hybrid recommender systems for electronic commerce. 2000.

[113] George Trigeorgis, Konstantinos Bousmalis, Stefanos Zafeiriou, and Bjorn W. Schuller. A deep matrix factorization method for learning attribute representations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(3):417–429, mar 2017.

[114] Udemy. Annual report pursuant to section 13 or 15(d) of the securities exchange act of 1934. 2023.

[115] Pranav Waila, Vivek Kumar Singh, and Manoj Kumar Singh. A Scientometric Analysis of Research in Recommender Systems. 5(1):71–84.

[116] Hao Wang. Alleviating sparsity problem of recommender system with no extra input data. *2022 5th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*, pages 691–695, 2022.

[117] Hao Wang. Dotmat: Solving cold-start problem and alleviating sparsity problem for recommender systems. *2022 IEEE 5th International Conference on Electronics Technology (ICET)*, pages 1323–1326, 2022.

[118] Qinqin Wang, Diarmuid O'Reilly-Morgan, E. Tragos, N. Hurley, B. Smyth, A. Lawlor, and Ruihai Dong. Learning domain-independent representations via shared weight auto-encoder for transfer learning in recommender systems. *IEEE Access*, 10:71961–71972, 2022.

[119] Yong Wang, Jiangzhou Deng, Jerry Gao, and Pu Zhang. A hybrid user similarity model for collaborative filtering. *Information Sciences*, 418, 08 2017.

[120] Hao Wu, Kun Yue, Xiaoxin Liu, Yijian Pei, and Bo Li. Context-aware recommendation via graph-based contextual modeling and postfiltering. *International Journal of Distributed Sensor Networks*, 11(8):613612, 2015.

[121] Shiwen Wu, Yuanxing Zhang, Wentao Zhang, Kaigui Bian, and Bin Cui. Enhanced review-based rating prediction by exploiting aside information and user influence. *Knowledge-Based Systems*, 222:107015, 2021.

[122] Kalidas Yeturu. Chapter 3 - machine learning algorithms, applications, and practices in data science. In Arni S.R. Srinivasa Rao and C.R. Rao, editors, *Principles and Methods for Data Science*, volume 43 of *Handbook of Statistics*, pages 81–206. Elsevier, 2020.

[123] Fuguo Zhang, Shumei Qi, Qihua Liu, Mingsong Mao, and A. Zeng. Alleviating the data sparsity problem of recommender systems by clustering nodes in bipartite networks. *Expert Syst. Appl.*, 149:113346, 2020.

[124] Qian Zhang, Jie Lu, and Guangquan Zhang. Cross-domain recommendation with multiple sources. *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2020.

[125] Weinan Zhang, Tianming Du, and Jun Wang. Deep learning over multi-field categorical data: A case study on user response prediction. *CoRR*, abs/1601.02376, 2016.

[126] Lei Zheng, Vahid Noroozi, and Philip S. Yu. Joint deep modeling of users and items using reviews for recommendation. *CoRR*, abs/1701.04783, 2017.

[127] Hongde Zhou, Fei Xiong, and Hongshu Chen. A comprehensive survey of recommender systems based on deep learning. *Applied Sciences*, 13(20), 2023.

[128] Andreas Zimmermann, Andreas Lorenz, and Reinhard Oppermann. An operational definition of context. volume 4635, pages 558–571, 08 2007.

[129] Erion Çano. Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, 21:1487–1524, 11 2017.