Tampere University

Uzair Ahmed

MULTI-ATTRIBUTE BIAS MITIGATION IN RECOMMENDER SYSTEMS

Faculty of Information Technology and Communication Sciences
M. Sc. Thesis
December 2023

ABSTRACT

Uzair Ahmed: Multi-Attribute Bias Mitigation in Recommender Systems
M.Sc. Thesis
Tampere University
Master's Degree Programme in Signal Processing and Machine Learning
December 2023

An abundance of data in this day and age opens up many opportunities for data consumption through many outlets. This calls for a framework to optimally categorize relevant data to be suggested to the users. Recommender systems have been in the limelight for quite some time now for this exact use case.

Variational Autoencoder (VAE) based recommender systems have been very successful in matching users with potentially relevant items. VAEs work on the assumption that similar user profiles have similar likenesses and behavior and can be suggested items by finding a pattern in their item relevancy. User profiles can be grouped up based on various factors, the most important one being their history of item rankings, and other personal attributes such as age, country, and sex. An optimal output from the VAE should take into account the most relevant items from the user groups but can also develop a bias towards their attributes which has to be mitigated or it can propagate as the data increases, i.e. learning a pattern that deduces that a certain nationality finds a certain item relevant, which can add unfairness and bias in the results.

A VAE-based framework (Stefanidis & Borges, 2022) was developed to tackle this problem and minimize user unfairness and bias in recommendations by actively taking into account a user's sensitive attribute while training. We propose an improvement to this framework by taking into account multiple sensitive attributes at a time to minimize user unfairness as much as possible and improve the PREC@1 performance metric. In the end, we compare the results of our improved framework to F2VAE and document our findings through multiple metrics like precision (PREC), unfairness (UFAIR), normalized discounted cumulative gain (NDCG), and recall.

Keywords: Recommender Systems, Variational Autoencoder, Bias, Unfairness, Precision, Recall, Normalized Discounted Cumulative Gain.

Contents

1. Introduction....1
2. Literature Review....3
2.1 Recommender Systems....3
2.1.1 Collaborative Filtering....3
2.1.1.1 User-Based Collaborative Filtering....4
2.1.1.2 Item-Based Collaborative Filtering....4
2.1.2 Content-Based Filtering....5
2.1.3 Hybrid Filtering....6
2.1.4 Fairness in Recommendations....7
2.1.4.1 Relation between Fairness and Other Concepts....9
2.2 Neural Networks....10
2.2.1 Backpropagation....11
2.2.2 AutoEncoders....13
3. Methodology....15
3.1 Variational AutoEncoders....15
3.2 F2VAE....18
3.3 Multi-Sensitive Attribute Bias Mitigation....19
4. Evaluation....21
4.1 Data....21
4.1.1 Data Preprocessing....23
4.1.2 Formalization....23
4.2 Evaluation Metric....24
4.3 Experiments....26
4.3.1 Results....28
5. Conclusion....33
References....34

1. Introduction

The exponential increase in data consumption and its widespread availability among the masses has given rise to the indispensability of recommender systems. As individuals generate and interact with vast amounts of data through various online platforms, the need for efficient and personalized content recommendations has become more pronounced. Social media platforms like Facebook and Instagram, streaming services such as Netflix and Spotify, and e-commerce giants like Amazon have become data-rich environments, collecting information on user preferences, behaviors, and interactions. Recommender systems utilize this wealth of data to analyze patterns, predict user preferences, and provide tailored recommendations, enhancing user experience and engagement. For instance, Netflix employs sophisticated algorithms to suggest movies and TV shows based on viewing history, while Amazon suggests products based on past purchases and browsing behavior. The ubiquity of data and the demand for personalized experiences underscore the increasing necessity of recommender systems in today's digital landscape. Recommender systems aim to enhance the user experience in many domains by linking the user with relevant data with ease (Roy & Dutta, 2022). Many modern recommender systems work on the assumption that the content consumed by a user group can be recommended to a similar user group. The recommendation is made based on profile similarity between different users. Two common techniques working with these concepts are Collaborative Filtering (Zhang, 2014) and Content-based Filtering (Zisopoulos, n.d.). An example is the movie streaming platform, Netflix, which recommends movies to users based on what other similar users have watched or found interesting.

Collaborative filtering is a popular technique in recommender systems that relies on user collaboration and preferences to make personalized recommendations. This approach assumes that users who have similar tastes in the past will likely have similar preferences in the future. There are two main types of collaborative filtering: user-based and item-based. In user-based collaborative filtering, recommendations are made based on the preferences of users with similar tastes, while item-based collaborative filtering recommends items similar to those the user has liked before. This technique has proven effective in various applications, from e-commerce platforms to streaming services (Breese et al., 2013). Collaborative filtering has seamlessly integrated with deep learning methodologies, particularly through the incorporation of autoencoders. (Zhang et al., 2019) discusses the use of autoencoders for recommender systems. Autoencoder-based solutions typically take in a user interaction matrix and try to learn the underlying representation through the hidden layers. The ratings are usually sparse and the latent representation is learned during the encoding phase and the decoder tries to generate ratings based on the learned representations over many users. One such example is AutoRec (Sedhain et al., 2015).

Variational Autoencoders have emerged as a powerful tool for collaborative filtering tasks by enabling the modeling of intricate patterns in user-item interactions (Kingma & Welling, 2013). It provides a multinomial generative model which is useful to find latent-variable space that is non-linear. Advancements in the VAE domain saw a rise of many VAE-based neural networks for recommender systems, for example,

Collaborative Vaes (Li & She, 2017, 305-314) and Hybrid Vaes (HVAE) (Gupta et al., 2018). In the context of CVAE, the rating matrix's construction involves utilizing user-item embeddings that represent items in a latent space. In contrast, HVAE relies solely on variational autoencoders to reconstruct the complete user history.

Recommender systems rely on large amounts of data to train and function effectively and with every large data model, bias and unfairness finds its way into the system. Fairness is a crucial consideration in the development and deployment of recommender models, as these systems play a pivotal role in shaping user experiences and influencing decisions. Unfairness and bias in recommender models can significantly impact the results, leading to unequal treatment and potential harm to certain user groups. When recommender models exhibit bias, they may disproportionately favor specific demographics, preferences, or content, creating an imbalance in recommendations. This can result in limited exposure to diverse perspectives and the reinforcement of existing stereotypes. Moreover, biased recommendations may contribute to systemic inequalities and hinder the potential for discovering new content or products. Therefore, addressing fairness concerns in recommender models is imperative to ensure that users receive equitable and unbiased recommendations, promoting a more inclusive and representative online environment. Ongoing efforts in research and industry aim to develop techniques and frameworks that mitigate biases in recommender systems and enhance their fairness. ALFR-VAE (Edwards & Storkey, 2015) tries to minimize the effect of sensitive attributes like demographic information on the latent representation learned by the encoder for a fairer representation. It tries to predict a user-sensitive attribute using a binary classifier from the latent representation. FaiRVAE introduces a new error term for mitigating user bias and concatenating the error in the overall loss function of a VAE. F2VAE (Stefanidis & Borges, 2022) combines this implementation to minimize user bias and increase the long-term fairness of the recommender results.

Our primary focus in this research is to refine and enhance the existing F2VAE recommender system model, with a particular emphasis on improving sensitivity to user diversity, personalization, and addressing biases related to sensitive attributes. The core problem we are tackling involves the limitations of the original model which relies on a binary classifier to predict a single sensitive attribute. The binary classifier might oversimplify user preferences. A user behavior pattern might significantly be influenced by a group of sensitive attributes and the attributes might also be correlated. We investigate whether replacing this binary classifier with a more sophisticated and deeper multi-label classifier can effectively capture a broader range of user characteristics, providing a more nuanced representation of diverse attributes. The investigation into the efficacy of a multi-label classifier aligns with the broader goal of enhancing sensitivity to user diversity, improving personalization, and promoting fairness within recommender systems. Through this research question, the study seeks to contribute to the advancement of recommender system architectures, ultimately striving for more accurate, personalized, and equitable user experiences. Our method is aimed at maximizing the classification loss associated with multiple sensitive attributes, with the goal of strengthening the adversarial training mechanism and advancing the recommender system's capabilities in terms of relevancy, personalization, and fairness. The integration of a deeper multi-label classifier in our recommender system model represents a significant stride towards maximizing the classification loss associated with

multiple sensitive attributes. The enhanced depth of the classifier enables a more intricate analysis of user profiles by considering a broader spectrum of attributes simultaneously. This depth is instrumental in capturing the complexities inherent in user characteristics, as users often exhibit a diverse array of sensitive attributes. By accommodating a multitude of attributes, our classifier aims to generate comprehensive and nuanced predictions of user traits. Consequently, the increased granularity facilitates the amplification of classification loss for multiple sensitive attributes, fostering a more robust adversarial training mechanism within the Variational Autoencoder. This advancement aligns with the overarching objective of minimizing the influence of sensitive attributes on the latent space, reinforcing user privacy, and promoting fairness in the recommender system.

We train our proposed implementation on the same parameters as the existing F2VAE implementation to measure the direct impact of our changes on the results. We use the LFM-2B dataset with more than 2 billion listening events from users. The dataset is instrumental in our research as it is the only one in our knowledge with user demographic information present. We use a couple of metrics to evaluate and compare our results, namely precision@k (PREC@k), recall@k, unfairness@k (UFAIR@k) and NDCG@k. The precision@k, recall@k, ndcg@k, and UFAIR@k metrics collectively provide a thorough evaluation framework for the recommender system. Precision@k gauges recommendation accuracy in the top-k list, reflecting the proportion of correct suggestions. Higher precision@k values indicate more accurate recommendations aligned with user preferences. Recall@k assesses the system's ability to retrieve relevant items within the top-k recommendations, offering insight into the coverage of user preferences. Ndcg@k considers both relevance and ranking, emphasizing the importance of well-ordered recommendations. Elevated precision@k, recall@k, and ndcg@k scores collectively signify accurate, diverse, and well-ordered suggestions. Additionally, the UFAIR@k metric introduces a focus on fairness, specifically capturing unfairness in the top-k recommendations. A lower UFAIR@k score suggests reduced bias and improved fairness in the recommender system's outputs, contributing to a more equitable user experience. In essence, these metrics collectively provide a comprehensive assessment of the recommender system's effectiveness, accuracy, fairness, and overall optimization of the user experience.

Our contribution can be summarized as extending the F2VAE implementation and introducing a better classifier for the latent vector which can handle multiple sensitive attributes, and measuring the amount of improvement in fairness and quality of recommendations and the reduction in user bias.

2. Literature Review

This chapter discusses the advancements made in the field of recommender systems starting from the earliest adaptations to modern state-of-the-art neural implementations. The discussion also includes topics related to types of recommender systems and fairness in recommendations.

2.1 Recommender Systems

The idea of a system that recommends relevant items to a user based on a history of similar interactions with other groups of people can be traced back to the implementation by Elaine Rich in the 1970s (Beel, 2016,). Her idea was to ask a user several questions and then assign the user a group which then assumes a set of stereotypes about the user and recommends them a book they might like.

The landscape has changed since then. With the explosion of data and digitalization, the demand for better recommender systems increased drastically which in turn saw the rise of research on recommender systems, and many different types and implementations of such systems were seen. Almost every aspect of our digital consumption is backed by a recommender system in one way or another. Explained simply, recommender systems have become an essential part of our digital structure (Polatidis, 2013).

Diving deeper into the basic functionality of a recommender system, let us take an example of a recommender system with many users U which are associated with several items I. Each user has potentially rated several items from the list of items I and the relation can be visualized by a matrix of the shape (U, I) where a single item can be represented by (u, i) where $u \in U$ and $i \in I$. The matrix (U, I) can be sparse as not every user has rated every item. This is where a recommender system comes in. A recommender system estimates ratings for items that were initially not rated by a user by finding relevance between user groups through the items they had already rated.

Over the years many different techniques for recommender systems have been explored and studied with each having its pros and cons. The most famous ones are collaborative filtering (Zhang, 2014), content-based filtering (Zisopoulos, n.d.), and hybrid filtering (Çano & Morisio, 2019).

2.1.1 Collaborative Filtering

Collaborative filtering was first coined by a team of researchers when they developed a filtering system in 1992 (Goldberg et al., 1992). The recommender system domain has ever since spearheaded forward based on this technique. The method works off of the assumption that if user A has the same interests as user B, A is more likely to have a similar profile as B. Collaborative filtering is a process that involves looking for patterns of interests in a group of users (collaborating). Recommendations are then generated by generating a similarity matrix between the group and matching users with similar scores (Herlocker et al., n.d.).
Typically, Collaborative Filtering (CF) utilizes a user-item matrix with dimensions (U, I), where an item rating can be denoted as (u, i), signifying that u is an element of the user set U, and i is an element of the item set I. User ratings are compared, and predictions for unrated items are made based on the ranking of similar users. The collaborative effort of similarly ranked users influences the optimal outcome, generating new ratings that align with a user's interests.

Among the various approaches for defining the similarity matrix and computing ratings for a new item, the most prevalent ones are user-based and item-based CF (Shi et al., 2008).

User-Based: Recommending items to a user that other similarly matched users have rated highly.

Item-Based: Predicting items that a user might like based on the ratings given to similar items by the user.

2.1.1.1 User-Based Collaborative Filtering

User-based collaborative filtering, also known as K nearest neighbors filtering, works on the premise of finding other users with a similar rating pattern as the user who needs an item to be recommended (Ekstrand, 2011). To recommend a new item to user A, user-based CF looks for users with a rating history similar to user A. This history of ratings is then used to predict the rating for a new item previously not rated.

Besides the user-item matrix *(U,I),* a function *s:U×U'* is required to generate similarity scores between users and a technique to generate new ratings using this score.

To compute the rating *R,* for an item *i* for a user *u,* a weighted average of the neighbors is calculated taking similarity scores as the weight (Ekstrand, 2011)

$$R_{u,i} = \overline{R_u} + \frac{\sum_{u'}^{N} s(u,u')(R_{u,i} - \overline{R_{u'}})}{\sum_{u'}^{N} |s(u,u')|} \qquad \textbf{(2.1)}$$

2.1.1.2 Item-Based Collaborative Filtering

Item-based CF technique calculates the rating for a new unrated item by a user *u* by finding out the similarity between the items already rated by the user *u* (Sarwar et al., 2011). The similarity scores between items *i* and *i'* are found using similarity metrics such as Pearson similarity, conditional probability, or cosine similarity (Ekstrand, 2011).

$$sim(i,i') = \frac{\sum_{u}^{N}(r_{u,i} - \overline{r_u})(r_{u,i'} - \overline{r_u})}{\sqrt{\sum_{u}^{N}(r_{u,i} - \overline{r_u})^2}\sqrt{\sum_{u}^{N}(r_{u,i'} - \overline{r_u})^2}} \qquad \textbf{(2.2)}$$

Collaborative filtering shares its own sets of advantages and disadvantages. The biggest advantage of this technique is it does not need any prior knowledge about the domain it is being trained on which makes it suitable for many cases since all that is required is a set of users and the ratings they have provided for any set of items. The technique can also help find users new items that may be relevant to similar users.

The technique also suffers from a few issues, mainly the cold start issue and sparsity issue. The technique cannot recommend new items in the sense of that item not being seen in the training data of any user. The amount of data required for a stable recommender system is also large since a lot of data is needed for a CF technique to perform optimally (Lika et al., n.d.).

2.1.2 Content-Based Filtering

Content-based filtering technique takes into account users' history of choices and similarity count to recommend items to users. The recommendations are improved by taking the metadata about items into context. For example, suppose a user often watches movies with similar descriptions or opens web pages with a specific keyword appearing in them. In that case, a content-based filtering method will recommend items that have the highest count keyword from a user in its metadata (Thorat et al., n.d.). A keyword's importance in a text can be calculated using a metric commonly known as *tf*idf* weight (term-frequency times inverse document frequency). The weight of a term, denoted as *w(t,d)*, in document *(d)* within the *tf*idf* framework is determined by the term's frequency in the document (tf_t,d), the total number of documents containing the term (df_t), and the overall number of documents in the collection *(N)*.

$$w(t,d) = \frac{tf_{td} log(\frac{N}{df_t})}{\sqrt{\sum_i (tf_{t_i,d})^2 log(\frac{N}{df_{t_i}})^2}} \qquad \textbf{(2.3)}$$

Content-based recommendation systems make use of the item metadata to find relevant items to suggest to a user. It generates a distribution from a user profile and recommends items with a probability associated with them. Given a set of items to be ranked and several users, the technique computes the probability of an item being found relevant by a user (Pazzani & Billsus, n.d.). Many different types of machine learning algorithms can be used to generate relevant probabilities for an item like Decision Trees, Nearest Neighbor Methods, and Naive Bayes.

The working of a content-based filtering system can be stated in the following steps:

- Create a user profile based on the user's history of likes and dislikes.
- Deduce the attributes of items being used for recommendation.
- Compare the attributes of the item with the user profile/preferences.
- Recommend top-K items according to the features that match the user profile.

A content-based filtering technique overcomes the limitations of collaborative filtering by providing recommendations for entirely new items. Since this model can discern a user's specific interests, it can suggest highly niche items that only a minority of users may find appealing. However, this approach has its own set of drawbacks. The effectiveness of the technique relies heavily on the quality of representations used for text features and demands substantial domain knowledge to represent text meaningfully. Due to its reliance on a user's existing interests, the model's capacity to broaden a user's interests is notably restricted, a phenomenon known as the over-specialization issue (Stitini et al., 2023). Content-based filtering also grapples with the synonym issue, wherein it cannot differentiate between two or more words conveying the same meaning. Proposed solutions, such as the utilization of genetic algorithms and

Probabilistic Neighborhood Selection (PNS) algorithms, aim to address or mitigate these referencing challenges (Stitini et al., 2023).

2.1.3 Hybrid Filtering

The hybrid filtering technique overcomes the shortcomings of previous implementations by combining the models and generating recommendations (Hulbatte et al., 2018). The methods can be used together to generate an output or the algorithms can be implemented separately and then their outputs combined. The combination will produce more accurate, relevant, and general results. The combination can be done in a multitude of ways as defined by (Burke, 2002):

Weighted: A weighted hybrid recommender system takes into account the output of all the different recommendation algorithms and the result is computed by calculating the final rating by adding a weight to the output of each technique and combining them. An example would be the linear combination of all the previously discussed techniques. The advantage of such a system is that it can be fine-tuned by adjusting the weights given to all the parent techniques while maximizing the accuracy.

Switching: A switching hybrid system makes use of a predefined threshold or criteria to switch between the recommendation techniques. For example, a system can be built with collaborative filtering and content-based filtering, and the recommendation can be obtained from any of the other systems if the first system cannot generate an output with sufficient confidence, i.e. cannot meet the criteria or a threshold.

Mixed: A mixed hybrid system is recommended in cases where a large number of simultaneous recommendations are needed. Such a system can be built by utilizing different techniques and keeping the best of both worlds. A system can use collaborative filtering to generate relevant results while also suggesting new items to a user based on the output of a content-based filtering system. A mixed system concatenates the results of the techniques being used to provide a large number of recommendations to users.

Feature Combination: A feature combination recommender system combines the output from a collaborative filtering system and uses it as additional features for the input to a content-based filtering system. It lets the system have information on inherently similar items previously invisible to a plain CF system.

Cascade: Cascade Hybrid introduces a strict hierarchical structure to enhance recommendation systems, where the primary recommendation system produces the initial result. The secondary model then comes into play to address minor issues in the primary result, such as resolving ties in the scoring. This two-step approach proves especially effective in handling sparse datasets, offering solutions to equal scoring problems or missing data, ultimately contributing to more robust recommendations.

Meta Level: Meta-level hybrid shares similarities with the feature augmentation hybrid in that the contributing model enriches the main recommendation model with an augmented dataset. However, in contrast to the feature augmentation hybrid, the meta-level approach goes a step further by substituting the original dataset with a

learned model derived from the contributing model as the input to the main recommendation model.

Studies have shown that a hybrid recommendation system is very successful at generating relevant and accurate results. Netflix is a good example of a use case where a hybrid system has proven to be very accurate. It makes use of content-based filtering to recommend new items to users that are still relevant and collaborative filtering to suggest similar groups of people with movies. Furthermore, the positive results have been well documented by (Burke, 2002) in his research.

2.1.4 Fairness in Recommendations

A variety of data-driven systems exist in today's day and age and as the quantity of data increases, a system can unintentionally incorporate human biases or even introduce new ones into its results, and recommendation systems are not invulnerable to these defects (Chouldechova & Roth, 2020). An example can be an image search with the keyword ‘engineers’. The percentage of male representation in the images is a perfect example of stereotype exaggeration which is due to human bias. A quality recommender system should maintain a balance between representing different user groups and still producing novel results to expand a user's profile, it is unfair to ignore the wishes of a certain user group but the lack of newness is equally unfair in recommendations. A job posting is a perfect example of such a scenario where a user with low attributes related to the job will often click on low-salary jobs and will be recommended as such by the system due to his history (Leonhardt et al., 2018). Most approaches to algorithmic fairness wish to negate any discrimination in the output due to the attributes that are not relevant to the task at hand, the attributes are usually referred to as sensitive attributes (Pitoura et al., 2021). (Borges & Stefanidis, 2022) propose a way of mitigating bias and unfairness without any sensitive attribute information present in the data.

(Leonhardt et al., 2018) examines two primary sources of unfairness within recommendation systems. The first stems from constantly changing distributions resulting from items recommended to users, leading to infrequent recommendations for certain items. The second source arises from post-processing algorithms addressing marketplace unfairness. A substantial portion of the existing research concentrates on improving two fundamental aspects: (1) individual diversity, aiming to provide users with a range of recommendations, and (2) aggregate diversity, focused on diversifying recommended items across all users. Despite the intentions of individual and aggregate diversity to promote fairness for users and items, they do not address other dimensions of fairness, such as the differential treatment of users or items. For instance, in the pursuit of enhancing aggregate diversity, an online store might exclusively suggest top-rated items to a specific group, like wealthier users, while introducing new items (whose quality is uncertain) solely to users with lower income. This approach may result in unfairness to the latter group and simultaneously create a disparity by presenting new items only to users less likely to purchase them. Therefore, when devising fairness measures, it is crucial to incorporate criteria that avoid any biased treatment against specific user groups.

From the viewpoint of users, an equitable recommender system has the potential to amplify the presentation of diverse information within recommendations, potentially

introducing niche information or items to users. This, in turn, could contribute to broadening users' perspectives and augmenting the overall value of the recommendations. On the item side, a fair recommender system strives to sustain visibility for popular items while simultaneously encouraging the promotion of niche items, thereby fostering increased diversity and creativity within the recommended items. (Wang et al., 2023). (Borges & Stefanidis, 2021) proposed a way of mitigating popularity bias by having a weight parameter to control the penalty for popular items which enhances fairness from the items side.

(Wang et al., 2023) reference several definitions for fairness depending on their focus:

- **Process Fairness:** Process fairness concerns whether the recommendation model is fair and does not incorporate in any way, direct or indirect, unfair features (e.g., race, age, sex) while learning the representations.
- **Outcome Fairness:** Outcome fairness concerns whether the output of a recommendation system is fair, for example, are users recommended movies based purely on their preferences and not based on stereotypes based on sex.
    - **Grouped by Target:** Depending on whether the emphasis is on achieving fairness at the group level or for individuals, outcome fairness can be further classified.
        - **Group Fairness:** Various methods exist for categorizing groups, with gender, age, race, and country being the most prevalent attributes associated with fairness. Group fairness pertains to ensuring equitable outcomes among diverse groups and subgroups.
        - **Individual Fairness:** Individual fairness refers to the idea that similar individuals should be treated similarly.

2.1.4.1 Relation between Fairness and Other Concepts

In this subsection, we discuss the relationship between the concept of fairness and a few related concepts in the context of recommender systems.

Bias: Bias and every data-driven system go hand in hand and can exist in the data, the models, and the results. Data bias can come from data collection to preprocessing and can be categorized into *statistical bias* and *pre-existing bias* (LI et al., 2022).

- **Statistical Bias:** Statistical bias occurs when there are flaws in the design or the data collection process, that is when the data does not truly represent the population. For example, the selection is not random from a full population, wrongly preprocessed, or missing, resulting in a lack of diversity and other anomalies. Statistical bias can easily lead to unfairness in the process.
- **Pre-existing Bias:** Bias can exist even if the data is collected and preprocessed fairly. This happens when the data itself conveys a distorted image and biased decisions, which leads to a system that is not objective

and fair. An example can be a company that is known for hiring people of a certain attribute and they wish to train a model that does the hiring for them based on their data. The learned system will have a systematic bias towards people possessing that specific attribute which will cause unfairness. One example of pre-existing bias can be popularity bias which recommender systems are usually very susceptible to. The training is very much influenced by popular items in the data.

Bias increases unfairness in the results and the process. For example, (Zhu et al., 2021) demonstrated that matrix factorization models suffer from popularity bias in the process. Bias in data may tend to learn some sensitive information that is not relevant to the use case which increases process unfairness, which in turn increases unfairness in the outcome.

Diversity: Diversity in recommender systems concerns the diversity of items in the recommended list. Diversity goes hand in hand with fairness since it can affect user satisfaction. Increasing diversity can positively affect item fairness by recommending niche items from many categories while also giving popular items the correct exposure. Still, increased diversity does not necessarily improve item fairness.

The segment delved into numerous factors related to fairness in recommender systems and their respective roles. The fairness of a recommender system may be considered subjective, depending on its focus or use case. However, a fair recommender system can be broadly characterized by achieving an optimal equilibrium among various variables, including diversity, bias, and ranking. These approaches not only aid in comprehending the origins of unfairness but also pave the way toward unbiased recommendations.

2.2 Neural Networks

A Neural Network as apparent by its name is used to describe the connections in a brain. The idea was first presented by N. Rashevsky in 1938 (Rashevsky, 1938, 178-184). A brain consists of millions of these neurons which trigger based on inputs that are transported to them by the synapses. An activation triggered on one neuron is passed down to neurons ahead and it can be visualized as a graph. The design was further researched and described by McCulloch and Pitts (Mcculloch & Pitts, 1943).

An algorithmic representation of such a graph is well known today as an artificial neural network. An artificial neural network can contain many of these neurons which are called perceptrons in the algorithmic world. The term perceptron was coined by Frank Rosenblatt, a Ph.D. at Cornell, in 1958 when he developed the first machine that learned to distinguish between different types of punch cards (Rosenblatt, 1958, 386-408). A perceptron would take in inputs just like a neuron in the brain and give an output that could be used to act if the output was relevant.

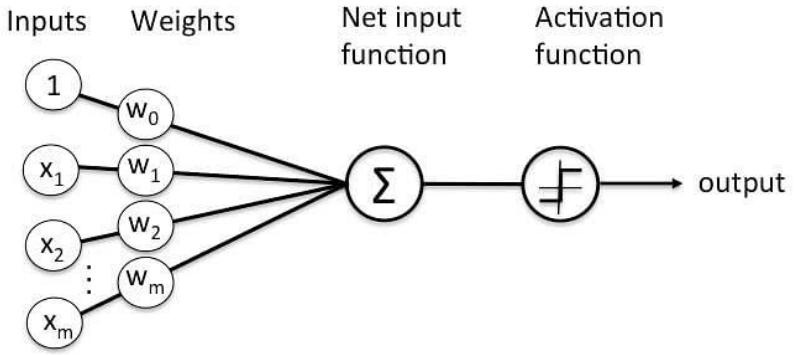

Figure 2.1 Perceptron

Each input, (x_j) carries a weight with it which is denoted as (W_{ij}) that is multiplied by its input and finally all the inputs are summed up to form a single value to be perceived by the perceptron as shown in Figure 2.1. The final output from a perceptron can be measured by passing the cumulative input to an activation function. Many different activation functions can be used based on the need. A few of the common ones are Sigmoid, ReLu, and Tanh. The difference lies in their output as the output of a sigmoid ranges between 0 and 1, ReLu ranges between 0 and infinity and Tanh ranges between -1 and 1. These activation functions are non-linear and are preferred mostly to introduce non-linearity in the model while linear functions exist as well.

The flow described in figure 2.1 can be summarized by equations 2.4 and 2.5 by using Sigmoid as the activation function σ (x_j).

$$x_j = \sum_{j \in i \; in \; connections} W_{i,j} x_i \qquad \textbf{(2.4)}$$

$$y_j = \sigma(x_j) = \frac{1}{1+e^{-x_j}} \qquad \textbf{(2.5)}$$

A typical ANN (Artificial Neural Network) consists of many perceptrons divided into layers. In a simple ANN, there are 3 layers namely, the Input layer, the Hidden Layer, and the Output Layer. The input layer represents the number of inputs being fed to the network and has the same number of nodes as the length of the input. The hidden layer consists of perceptrons which can range from 1 to as many as needed. The final output layer consists of perceptrons which output the final values. This architecture is also named a Multilayer Perceptron (MLP) (Stegemann & Buenfeld, 1999).

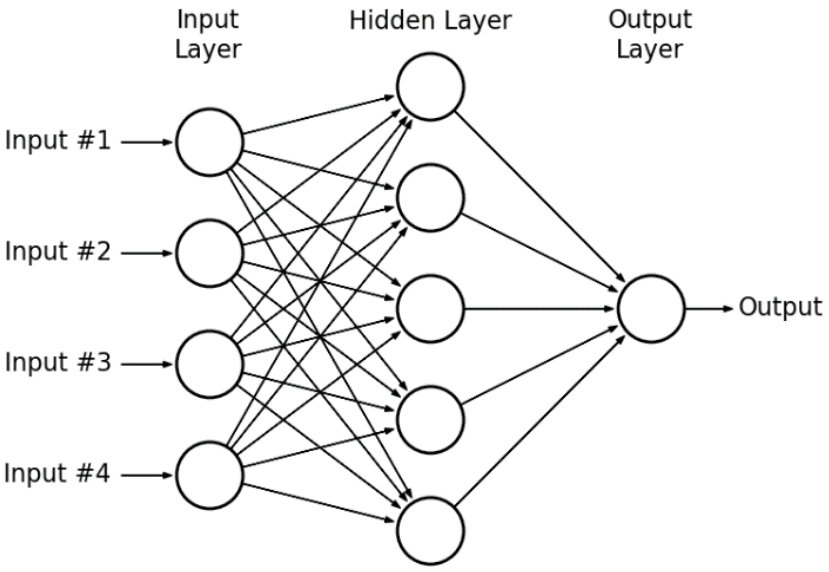

Figure 2.2 Multilayer Perceptron

As depicted in Figure 2.2, the information flow is in the forward direction. This type of ANN is also known as a feed-forward network. A feed-forward network's data flow is strictly limited to the forward direction and there are no feedback loops in the network. The configuration of an ANN can be changed, and multiple hidden layers can be added but the data flow is always going to be in the forward direction. Having no way to pass information backward, this simple implementation has no way of learning based on some metric. To learn distributions and patterns in the input data, a way has to be established to pass information backward to each of the nodes to update their weights in the correct direction and minimize the output error. This concept is known as backpropagation.

2.2.1 Backpropagation

A neural network tries to learn the pattern or the distribution hidden within the input data through a process called training. The training phase of a network usually comprises a loop that trains the network iteratively. In each iteration, the network generates the outputs which are then compared with the ground truth, and the error term is calculated and the error rate is back propagated in the network to adjust the weights of each neuron, or perceptrons. The weight tuning helps minimize the error as the training goes on and learns a more generalized representation of the input. The main idea of an ANN training phase is for the weights to learn and store the information about the data so that correct predictions can be made from an input term.

Backpropagation uses a method called gradient descent to find the right weights by minimizing the Error term as described by (Rumelhart et al., 1986, 533−536)

$$E = \frac{1}{2}\sum_{c}\sum_{j}(y_{j,c} - y'_{j,c})^2 \qquad \textbf{(2.6)}$$

where *c* is an index over input-output pairs, *j* is an index over output units, *y* is the output from the model and *y'* is the ground truth for an input.
We can assume a network by using Equation 2.5 as the output activation function and Equation 2.4 as the cumulative inputs for all the hidden layers. The backward pass starts by computing $\frac{\delta E}{\delta y}$ for each of the output units. We can differentiate equation 2.6 for a particular input-output pair *c,j,* and suppressing the index c gives us

$$\frac{\delta E}{\delta y_j} = y_j - y'_j \qquad \textbf{(2.7)}$$

We can then apply the chain rule to compute $\frac{\delta E}{\delta x_j}$

$$\frac{\delta E}{\delta x_j} = \left(\frac{\delta E}{\delta y_j}\right)\left(\frac{dy_j}{dx_j}\right)$$

Differentiating equation 2.5 and then substituting to get the value of $\frac{dy_j}{dx_j}$

$$\frac{\delta E}{\delta x_j} = \left(\frac{\delta E}{\delta y_j}\right) y_j (1 - y_j) \qquad \textbf{(2.8)}$$

This equation informs us about the impact of a variation in the input, x, on the error. However, the overall input is merely a linear function of the states from all preceding lower levels, and it also constitutes a linear function of the weights of the connections. Consequently, it is straightforward to calculate how altering the weights will influence the error.

For weight $w_{i,j}$ from *i* to *j* the derivative is as follows

$$\frac{\delta E}{\delta w_{j,i}} = \frac{\delta E}{\delta x_j}\frac{\delta x_j}{\delta w_{j,i}}$$

$$\frac{\delta E}{\delta w_{j,i}} = \frac{\delta E}{\delta x_j} y_i \qquad \textbf{(2.9)}$$

and for the output of the i^{th} unit, the contribution $\frac{\delta E}{\delta y_j}$ resulting from the effect of *i* on *j* is simply

$$\frac{\delta E}{\delta x_j}\frac{\delta x_j}{\delta y_i} = \frac{\delta E}{\delta x_j} w_{j,i}$$

and taking into context, all the connections propagating back from unit *i*, we have

$$\frac{\delta E}{\delta y_i} = \sum_j \frac{\delta E}{\delta x_j} w_{j,i} \qquad \textbf{(2.10)}$$

We can now compute $\frac{\delta E}{\delta y}$ for any layer as long as we have $\frac{\delta E}{\delta y}$ for the next layer. This procedure can therefore be repeated to compute this term for all the layers computing the change in error due to the weights as we go. $\frac{\delta E}{\delta w}$ can be accumulated over all the input-output cases before updating the weights. A simple version of gradient descent is to change each weight by an amount proportional to the accumulated $\frac{\delta E}{\delta w}$ (Rumelhart et al., 1986, 533-536).

2.2.2 AutoEncoders

AutoEncoders is a type of feed-forward ANN that excels in learning an identity function in an unsupervised way that can compress and reconstruct the input. Doing this, it learns more efficiently and compresses the representation of the original input (Rumelhart & McClelland, 1987).

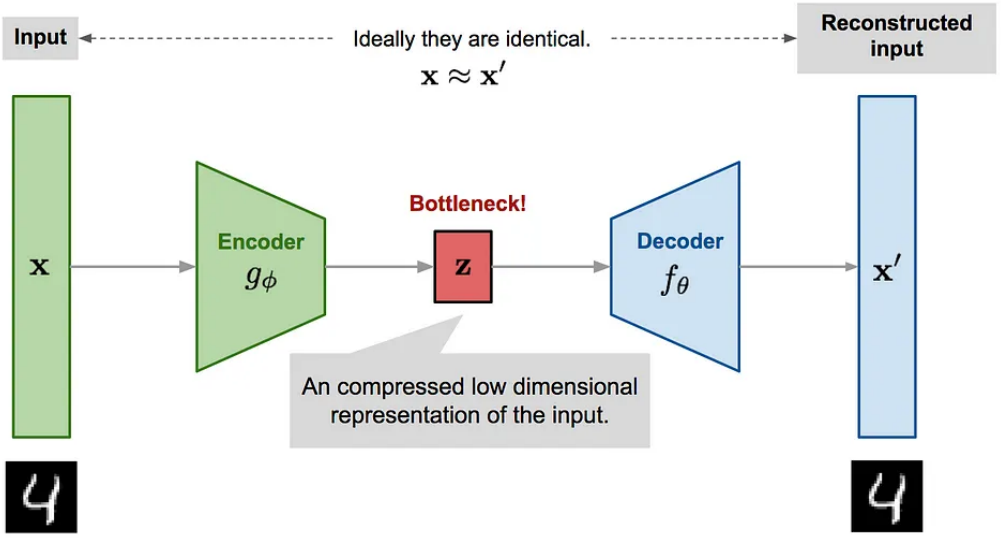

Figure 2.3 Autoencoder

Figure 2.3 portrays a typical autoencoder architecture. As visible in the figure, an autoencoder consists of three significant parts:

- **The Encoder Block g_ϕ:** The encoder block takes in the input vector and passes it through its hidden layers, compressing the input representation to compute *z*.

It is a simple feed-forward ANN that acts as a dimensionality reduction technique that learns a compressed representation of the input.

$$z = g_{\phi}(x) = Wx + b_{z} \qquad \textbf{(2.11)}$$

Where g_{ϕ} is the activation function for the encoder output, *x is the input vector, W* is the weight vector for all the connections and b represents their respective biases.

- **The Bottleneck:** The bottleneck represented by z is the compressed representation learned by the encoder block.
- **The Decoder Block f_{θ}:** The decoder block takes in the compressed representation computed by the encoder and tries to reconstruct the original input from it. The decoder ANN tries to learn the parameters required to reconstruct the compressed representation.

$$\hat{x} = f_{\theta}(g_{\phi}(x)) = W'z + b_{x} \qquad \textbf{(2.12)}$$

The reconstruction is represented by $\hat{x}$ and the reconstruction is compared by the original input to calculate the loss which can then be minimized by backpropagation. The autoencoder training step finds optimal values for the set of parameters *W, W',* $b_{z,,}$ and b_{x}. The loss is calculated using the mean squared error function represented as $L(x, \hat{x})$.

$$L(x, \hat{x}) = \frac{1}{n}\sum_{i=1}^{n}(x^{i} - f_{\theta}(g_{\phi}(x^{i})))^{2} \qquad \textbf{(2.13)}$$

To summarize, an autoencoder is an ANN that intends to reconstruct the input. The shape of the output is always equal to the input shape. An autoencoder is considered equal to Principal Component Analysis (PCA) (Maćkiewicz & Ratajczak, 1993, 303-342).

3. Methodology

This chapter focuses on modern implementations in the domain of recommender systems. New research and improvements have paved the way for better recommender systems which are more accurate, fair, and better generalized over a large group of subjects. Autoencoders have been part of many such enhancements. A new version of Autoencoder known as the Variational Autoencoder (VAE) has spearheaded the domain of recommender systems and collaborative filtering ever since. Data-driven domains are

always susceptible to unfairness issues and the system must be fair in its dealings and recommender systems are no exception. A recommender system must minimize user unfairness and bias while also giving relevant results, i.e. fair to all groups of users and negating the effects of their attributes on the recommendations. A user can have several sensitive attributes related to them like age, sex, country, etc. A framework must be developed that gives us the control to manage sensitive attributes actively while training regardless of how many they are if they exist in the training data. To address this we used the existing F2VAE model which mitigates user bias by adding two new loss terms and by taking into account a single sensitive attribute while training (Stefanidis & Borges, 2022).

3.1 Variational AutoEncoders

Variational Autoencoders (VAEs) blend autoencoders and probabilistic graphical models, introduced by Kingma and Welling in their 2013 paper (Kingma & Welling, 2013). VAEs learn a probabilistic mapping from input data to a latent space, treating latent variables as sampled from a Gaussian distribution. This probabilistic approach allows VAEs to generate diverse and realistic samples. Training involves maximizing a lower bound on data likelihood, encouraging the model to learn meaningful representations and a structured latent space.

VAEs are responsible for mapping the input *x* to a probability distribution which can be represented as p_θ. θ represents the parameters for the distribution. Generating a new sample that is close to the original input *x* can be labeled as a two-step process:

1. A latent variable *z* is generated from the prior distribution $p_\theta(z)$.
2. A sample *x'* is generated from a conditional distribution $p_\theta(x'|z)$.

The algorithm is only aware of the input *x* but the characteristics of the variable *z* are unknown which has to be inferred:

$$p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)}$$

where, $p_\theta(x) = \int p_\theta(x|z)p_\theta(z)dz$. This computation is intractable and to overcome, an approximation distribution $q_\phi(z|x)$ is introduced to approximate $p_\theta(z|x)$ (Blei et al., 2016). While approximating, we must ensure that the two distributions $p_\theta(z|x)$ are as similar as possible. Kullback–Leibler divergence (KL divergence) metric is introduced here which is a measure of similarity between two probability distributions, thus to ensure similarity, we could minimize the KL divergence between the two distributions. Minimizing the aforementioned metric results in the loss function for a typical VAE, also known as Evidence Lower Bound (ELBO).

ELBO Loss Function:

$$D_{KL}(q_\phi(z|x)||p_\theta(z|x))$$

$$= \int q_\phi(z|x) log \frac{q_\phi(z|x)}{p_\theta(z|x)} dz$$

$$= \int q_\phi(z|x) log \frac{q_\phi(z|x) P_\theta(x)}{p_\theta(z,x)} dz$$

$$= \int q_\phi(z|x) \left[log p_\theta(x) + log \frac{q_\phi(z|x)}{p_\theta(z,x)} \right] dz$$

$$= \int q_\phi(z|x) log p_\theta(x) dz + \int q_\phi(z|x) log \frac{q_\phi(z|x)}{p_\theta(z,x)} dz$$

$$= log p_\theta(x) + \int q_\phi(z|x) log \frac{q_\phi(z|x)}{p_\theta(x|z) p_\theta(z)} dz$$

$$= log p_\theta(x) + \int q_\phi(z|x) \left[log \frac{q_\phi(z|x)}{p_\theta(z)} - log p_\theta(x|z) \right] dz$$

$$= log p_\theta(x) + \int q_\phi(z|x) log \frac{q_\phi(z|x)}{p_\theta(z)} dz - \int q_\phi(z|x) log p_\theta(x|z) dz$$

$$= log p_\theta(x) + E_{z \sim q_\phi(z|x)} log \frac{q_\phi(z|x)}{p_\theta(z)} - E_{z \sim q_\phi(z|x)} log p_\theta(x|z)$$

$$= log p_\theta(x) + D_{KL}(q_\phi(z|x)||p_\theta(z)) - E_{z \sim q_\phi(z|x)} log p_\theta(x|z)$$

$$\Rightarrow log p_\theta(x) - D_{KL}(q_\phi(z|x)||p_\theta(z|x)) = E_{z \sim q_\phi(z|x)} log p_\theta(x|z) - D_{KL}(q_\phi(z|x)||p_\theta(z))$$

(3.1)

The L.H.S. of equation 3.1 consists of the log-likelihood of generating the real input and the negative KL divergence, these are the terms we wish to maximize.

The R.H.S. of equation 3.1 is called the lower bound (ELBO) as it is always

<= $log p_\theta(x)$. The loss function consists of the negative of this term as it is to be minimized (Kingma & Welling, 2013).

$$Loss(\theta, \phi) = - E_{z \sim q_\phi(z|x)} log p_\theta(x|z) + D_{KL}(q_\phi(z|x)||p_\theta(z)) \quad \textbf{(3.2)}$$

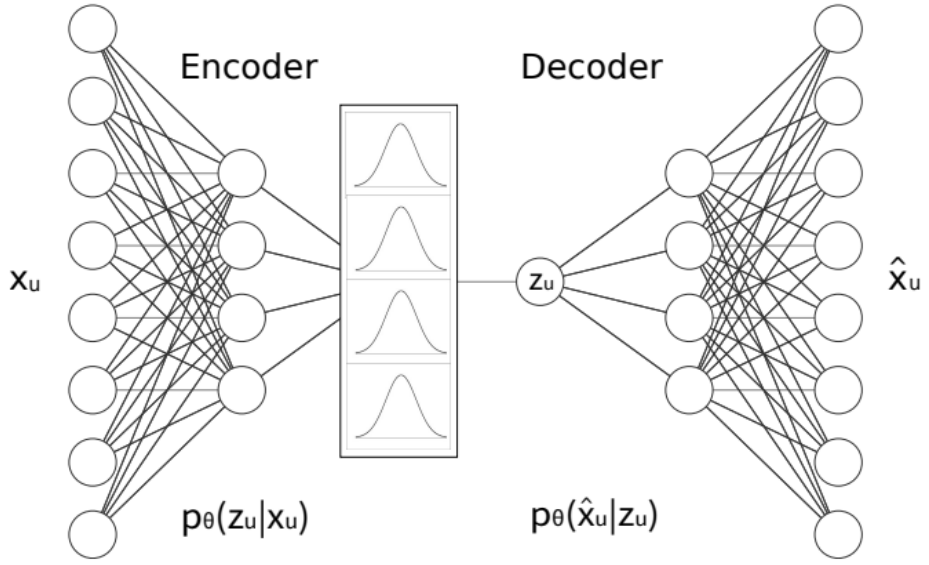

Figure 3.1 Variational Autoencoder (Borges and Stefanidis 2019)

Just like an autoencoder, a variational autoencoder also consists of an Encoder and a Decoder. The Encoder encodes an input x into two variables; one for the mean and the other for the standard deviation of the multivariate normal distribution of the latent space. The Decoder takes the mean and standard deviation of the distribution and is tasked with sampling the output with it according to $p_\theta(x|z)$ and the loss is calculated using equation 3.2. The sampling step blocks the gradients from flowing back into the encoder while backpropagating, hence the model will not train. To solve this issue, (Kingma & Welling, 2013) proposed a reparameterization trick.

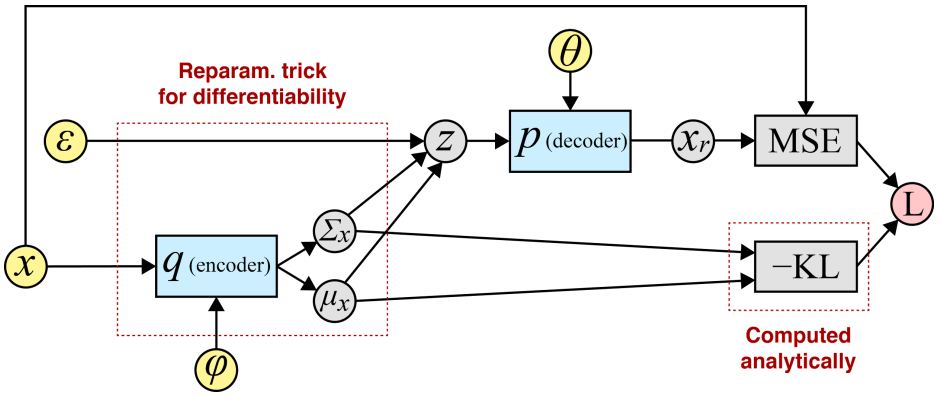

Figure 3.2 Reparameterization Trick

Rather than sampling z directly, we introduce a noise variable ϵ that is typically from a standard normal distribution and we use it to scale our predicted mean and variance which is then used to generate the latent variable z, which gives us:

$$z = \mu + \sigma \odot \epsilon, \; where \; \epsilon \sim N(0, I)$$

3.2 F2VAE

F2VAE is a framework presented by Borges and Stefanidis in 2022 (Stefanidis & Borges, 2022) dedicated to fair representation and fair recommendation. The framework builds upon ALFR-VAE (Edwards & Storkey, 2015) and FaiRVAE implementations. ALFR-VAE was originally proposed to reduce the influence of a binary user attribute in the context of fair classification. The idea was adapted to categorical sensitive attributes in the context of recommender systems in F2VAE. FaiRVAE introduces a new metric for measuring bias in the results and is focused on mitigating systematic differences between results offered to different groups. F2VAE combines these implementations to reduce unfairness according to a single loss function.

ALFR-VAE implements a binary log-loss for measuring the influence of a binary sensitive attribute in the latent vector *z*. F2VAE implements a categorical log-loss of a classifier network trained to predict the sensitive attribute *s* from *z*. The classifier is trained alongside the recommender system and the accuracy is calculated with

$$C_{\theta}(z, s) = - E\left[\sum_{i=1}^{N} s \cdot log(Pred_{i}(z))\right] \qquad \textbf{(3.3)}$$

where N is the number of sensitive attribute classes, $Pred_i$ as the Softmax probability for the i^{th} class. The cumulative loss function is now given by:

$$max_{\theta} min_{\phi} L_{\theta,\phi} = E_{\theta} + D_{\phi} + C_{\theta} \qquad \textbf{(3.4)}$$

The R.H.S. side of the equation partially comes from equation 3.2 presented earlier. The idea here is to maximize the classification loss to reduce as much sensitive information present in the latent vector *z*.

FaiRVAE introduces a new metric for mitigating user bias in recommendations expressed as:

$$F_{\phi}(u) = \left| \frac{1}{|U_s|} \sum_{c \in U_s} E\left[logp_{\theta}(x_c | z_c)\right] - \frac{1}{|U|} \sum_{u \in U} E\left[logp_{\theta}(x_u | z_u)\right] \right| \qquad \textbf{(3.5)}$$

where *s* represents the sensitive attribute related to the user *u*.
The final loss function then can be written as

$$max_{\theta} min_{\phi} L_{\theta,\phi} = E_{\theta} + D_{\phi} + C_{\theta} + F_{\phi} \qquad \textbf{(3.6)}$$

β, γ and τ hyperparameters are then added to the KL Divergence metric, classification loss, and the user unfairness metric to control their impact on the loss.

$$max_{\theta} min_{\phi} L_{\theta,\phi} = E_{\theta} + \beta \cdot D_{\phi} + \gamma \cdot C_{\theta} + \tau \cdot F_{\phi} \qquad \textbf{(3.7)}$$

The final loss function equation 3.7, E_{θ} represents the reconstruction error, D_{ϕ} represents the divergence between inner representations and the approximated distribution. These loss errors need to be minimized. C_{θ} stands for the prediction error for the classifier which tries to predict the class of sensitive attributes from the latent vector z which needs to be maximized. F_{ϕ} stands for user unfairness that needs to be minimized.

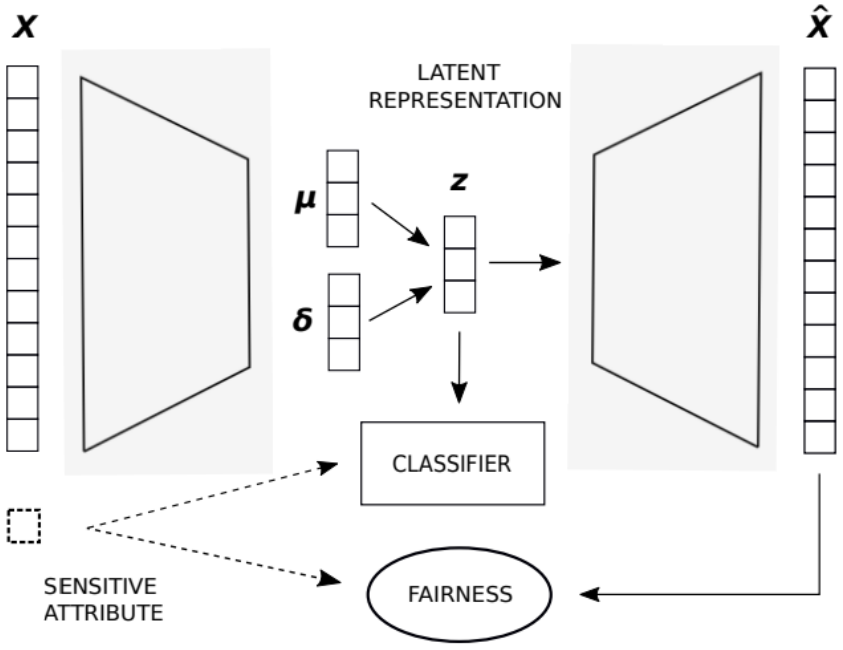

Figure 3.3: F2VAE architecture Stefanidis and Borges 2022

Gathering all these loss terms together would increase the chances for model weights to converge toward mitigating unfairness associated with a user-sensitive attribute. A general scheme can be inferred from Figure 3.3.

3.3 Multi-Sensitive Attribute Bias Mitigation

Recommender systems play a pivotal role in enhancing user experience by delivering personalized content, products, or services. However, existing models often overlook the intricate nature of user preferences and the potential influence of multiple sensitive attributes. Recommender systems, like F2VAE, have shaped how we get personalized recommendations. F2VAE uses a binary classifier in the Variational Autoencoder (VAE) set up to handle sensitive attribute inference. Despite its positive results, there's room to enhance this model, especially considering the evolving nature of user preferences and the need for a more detailed representation of diverse user traits.

This thesis aims to improve upon F2VAE by introducing a key upgrade. F2VAE's binary classifier simplifies user preferences, predicting a single sensitive attribute like a user's country. We propose moving from this binary approach to a more advanced multi-label classifier.

The first motivation is to boost the system's sensitivity to user diversity. While F2VAE tailors recommendations well, the binary classifier might oversimplify user preferences. By adopting a multi-label classifier, the improved model intends to capture multiple user-sensitive attributes at once, offering a richer understanding of user profiles.

Additionally, the shift to a multi-label classifier aims to enhance personalization beyond binary categories. F2VAE categorizes users into groups, potentially overlooking crucial aspects of their preferences. The multi-label approach promises a more detailed representation, allowing the recommender system to provide more accurate and finely tailored recommendations.

Concerning fairness and inclusivity, the move to a multi-label classifier is significant. F2VAE's binary approach may unintentionally introduce biases and fail to represent user diversity adequately. The proposed multi-label framework aligns with inclusivity and fairness principles, considering multiple sensitive attributes simultaneously for a more equitable user experience.

To strengthen the model against potential threats and protect user privacy, adversarial training within the VAE framework is maintained. The multi-label classifier contributes to adversarial robustness, ensuring the latent space remains resilient to sensitive attribute inference attempts.

We build upon the existing F2VAE framework (Stefanidis & Borges, 2022) and define the problem statement as using multiple sensitive attributes instead of just one and refactoring the classifier to a deeper multi-label classifier as opposed to a single layer classifier expecting a binary classification problem.

The classifier loss is changed from Equation 3.3 to

$$BCE\ loss = C_{\theta}(z,s) = -\frac{1}{N}\sum_{i \in U_s}^{N} s_i \cdot log(Pred_i(z_i)) + (1 - s_i) \cdot log(1 - Pred_i(z_i)) \quad \textbf{(3.8)}$$

where s_i corresponds to the one-hot encoded ground truth vector of the sensitive attributes related to a user, a general scheme of the ground truth vector is presented below

$$s_i = \begin{matrix} \text{US, FI, UK, m, n, ...} \\ [0, 1, 0, 1, 0 \ldots\ldots] \end{matrix}$$

where the example user belongs to Finland and is also a male. $\log(Pred_i)$ as log probabilities of the predicted classes identified from the latent representation z. $Pred_i(z)$ acts as the output probability function for the classifier which is defined as the sigmoid $\sigma(z)$ function presented below

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

The output shape of the function $Pred_i(z)$ matches that of the ground truth vector. Each element from the sigmoid output can be considered active since it is a multi-label classification problem and many sensitive classes can be attributed to a single user. The loss function is divided into two sections where the side on the left of the addition operator refers to when the ground truth label corresponds to a 1 and the right-hand side of the addition operator corresponds to the loss for when the ground truth is supposed to be 0. In a multi-label classification problem like ours, the BCE loss treats each individual label as a mini binary classification problem where the ground truth can be either 0 or 1. Both the positive and negative label errors are summed up and the total loss is averaged on the size of the ground truth vector. The negative sign is used to cancel out the negative values observed after applying the log function to probabilities.

The transition from cross-entropy loss in the existing implementation to binary cross-entropy loss in the proposed multi-label classification marks a substantial improvement in several aspects. Unlike traditional multi-class classification, binary cross-entropy loss excels in handling scenarios where each instance can belong to multiple classes simultaneously. This is particularly advantageous in our context of predicting multiple user-sensitive attributes concurrently. From a fairness perspective, the multi-label approach significantly contributes to mitigating bias by considering the simultaneous influence of multiple sensitive attributes. In contrast to the previous binary classifier focusing on a single aspect, the deeper multi-label classifier offers a more holistic view, reducing the risk of perpetuating biases associated with any single attribute. This shift aligns with the intention to foster fair and unbiased recommendations, promoting a recommender system that better respects the intricacies of individual user profiles. Using a deeper classifier network should in theory be able to better predict any sensitive information learned by the latent vector z and help mitigate any unfairness due to that. Also, having the ability to classify multiple sensitive attributes associated with a single user and maximizing the error should help mitigate a wider range of group biases and increase long-term fairness. We use a range of different sensitive attributes present in the dataset and compare the results with the original F2VAE model.

4. Evaluation

We used the LFM-2b dataset (Melchiorre et al., 2021) to evaluate our proposed multi-sensitive attribute incorporated model and compare the results to the default F2VAE model (Stefanidis & Borges, 2022).

4.1 Data

The dataset contains more than two billion listening events extracted from the LastFM platform. The dataset contains the listening histories of users from the past 15 years (from 2005 until 2020). This dataset to my knowledge is the only dataset that includes user demographic information alongside listening events. An overview of the data structure can be seen in Figure 4.1.

Field	Type	Meaning
User Id	Integer	Unique user Id
Country	String	Two-letter country code of the user
Age	Integer	Age of the user
Gender	String	Gender of the user as specified on Last.fm
Track Name	String	Name of the track
Artist Name	String	Name of the artist
Timestamp	Timestamp <YYYY-MM-DD 00:00:00>	Timestamp of the listening event

Figure 4.1: Dataset Structure

	user_id	country	age	sex	track_name	artist	timestamp
0	0	RU	18	m	Mr. Carter (Featuring Jay-Z)	Lil Wayne	2013-11-23 14:22:38
1	0	RU	18	m	Mr. Carter (Featuring Jay-Z)	Lil Wayne	2013-11-23 14:17:22
2	0	RU	18	m	Mr. Carter (Featuring Jay-Z)	Lil Wayne	2013-11-23 14:12:05
3	0	RU	18	m	Mr. Carter (Featuring Jay-Z)	Lil Wayne	2013-11-23 13:25:08
4	0	RU	18	m	Mr. Carter (Featuring Jay-Z)	Lil Wayne	2013-11-23 13:19:51

Figure 4.2: Dataset table view

Figure 4.2 gives a preview of the first five rows of the dataset. Each row represents a listening event from a user. A user can have multiple listening events associated with the same song, hence multiple rows with the same artist and track_name combination are possible.

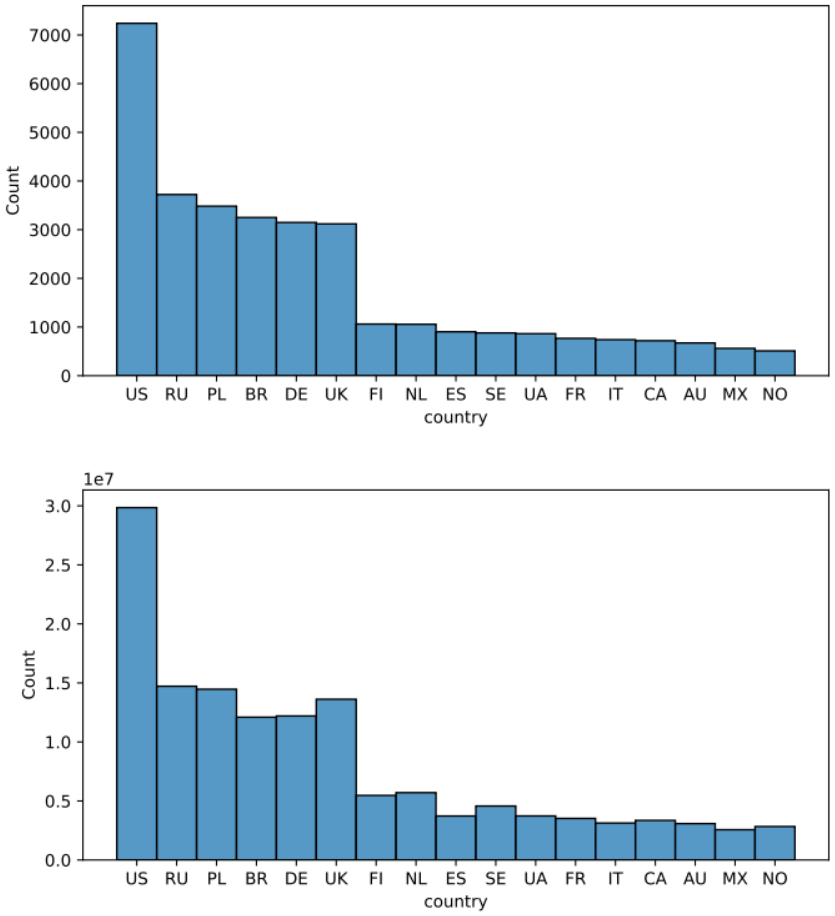

Figure 4.3: (Top) Number of users associated with each country in a subset. (Bottom) The number of interactions associated to each country in the same subset.

Figure 4.3 shows us the user count and listening event distribution in a subset of the total data. It is quite visible that most listening events originate from a single country and this trend carries over to the whole dataset.

4.1.1 Data Preprocessing

We first removed all the user records with missing demographic information. This is to ensure that all the data has associated sensitive attributes with it while training. We also removed all user records with less than 10 overall listening events and tracks with less than 100 overall listening events. This is to ensure mitigate any outliers in the data. It is then followed by dropping duplicate listening events to obtain binary relations between

users and tracks. To sum minority groups together, we kept the labels for countries with more than 500 users, and the other country labels were grouped in a single category.

The preprocessing gave us 16 country labels, and 3 sex labels and we also ended up with 30,073 users and 632,663 unique tracks. After this, user and track-specific data was saved to their respective files.

4.1.2 Formalization

The number of interactions for each user for each unique track is counted and the Users are split into train/validation/test subsets. All subsets are made with the shape *(U, I)* where the set *U* contains the users in the subset and *I* contains the total number of tracks. The values hold the number of interactions from a user for a track. A general schema can be inferred from Figure 4.4 below.

	track_name	artist	song_id
0	!	Böhse Onkelz	0
1	!	Mari! Mari!	1
2	!	Pato Fu	2
3	!	С.к.а.й.	3
4	! (Foreword)	Pain of Salvation	4

	user_id	age	sex	country
0	0	18	m	RU
1	1	18	m	PL
2	2	21	n	PL
3	3	18	m	OTHER
4	4	27	f	OTHER

Figure 4.4: (Left) Track data sample. (Right) User data sample

song_id	0	1	2	3	4	5	6	7	8	9	...	632653	632654	632655	632656	632657	632658	632659	632660	632661	632662
user_id																					
4382	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
23715	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
43864	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
29180	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
46741	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0

5 rows × 632663 columns

Figure 4.5: Sample training data

The users are also split into the query and ground truth subsets for validation and testing. When a recommendation comes through from the model, it is tested against the ground truth subset which contains relevant items that were withheld from the test/validation subset. These subsets are used for computing the performance of the model using the metrics defined below.

4.2 Evaluation Metric

We use several quality and fairness measures to evaluate the results of our proposed improvement versus the existing implementations.

- **Normalized Discounted Cumulative Gain (NDCG) @ *k*:** NDCG, or Normalized Discounted Cumulative Gain, is a widely used metric in recommender systems to evaluate the effectiveness of ranking algorithms. It considers both the relevance and the position of recommended items.
  The formula for NDCG is

$$NDCG@k = \frac{DCG@k}{IDCG@k} \qquad \textbf{(4.1)}$$

  Where:

  - DCG@*k* (Discounted Cumulative Gain at position *k*) is the sum of the relevance scores of the top *k* recommended items, discounted by their position.

$$DCG@k = \sum_{i=1}^{k} \frac{rel_i}{log_2(i+1)}$$

  - IDCG@*k* (Ideal Discounted Cumulative Gain at position *k*) represents the ideal ranking where items are sorted by relevance, and DCG is calculated for the top *k* items in this ideal ranking.

  NDCG@k values range from 0 to 1, with 1 indicating a perfect ranking where the most relevant items are at the top. It provides a normalized perspective by dividing the actual DCG by the IDCG to account for variations in the number of relevant items. Higher NDCG values indicate better recommendation quality.

- **Precision@k:** Precision@k is a metric commonly used in recommender systems to evaluate the accuracy of the top k recommendations. It measures the proportion of relevant items among the top k recommended items.

$$Precision@k = \frac{Number\ of\ relevant\ items\ in\ top\ k}{k} \qquad \textbf{(4.2)}$$

  Where "Number of relevant items in top k" refers to the count of items in the top *k* recommendations that are considered relevant based on a predefined criterion (e.g., user ratings, user interactions) and *k* is the number of items in the recommendation list.

  Precision@k values range from 0 to 1, where 1 indicates that all the top k recommendations are relevant. Precision@k provides insight into the ability of a recommender system to accurately suggest items that users find relevant within the top k positions.

- **Recall@k:** Recall@k is a metric used in recommender systems to evaluate the coverage of relevant items within the top k recommendations. It measures the proportion of relevant items that were successfully recommended out of all relevant items.

$$Recall@k = \frac{Number\ of\ relevant\ items\ in\ top\ k}{Total\ number\ of\ relevant\ items} \quad \textbf{(4.3)}$$

  Where "Number of relevant items in top k" refers to the count of items in the top k recommendations that are considered relevant based on a predefined criterion and "Total number of relevant items" is the overall count of the items that are relevant based on the same criterion.

  Recall@k values also range from 0 to 1, where 1 indicates that all relevant items are successfully recommended within the top k positions. A higher Recall@k suggests that the recommender system is effective at capturing a larger portion of relevant items in its recommendations.

- **Unfairness at k (UFAIR@k):** The UFAIR@k metric is a measure of disparity in the precision of top k recommendations across different user groups within a recommender system. It is calculated by summing the absolute differences between Precision@k (PREC@K) values for all pairs of user groups. The unfairness value can be further normalized by the number of comparisons as follows (Stefanidis & Borges, 2022)

$$UFAIR@k = \frac{1}{nxm} \sum_{n=1}^{|C|} \sum_{m=1}^{|C|} \left| \frac{1}{|C_n|} \sum_{u \epsilon C_n} PREC@k(u) - \frac{1}{|C_m|} \sum_{u \epsilon C_m} PREC@k(u) \right| \quad \textbf{(4.4)}$$

  Where set C contains all the groups that are formed by the sensitive attributes in the dataset selected for testing and $C_{n/m}$ is the subset of users associated with that group.

4.3 Experiments

The dataset is divided into train/test subsets as described earlier. All the models are trained on the training set and then evaluated on the test and their performance is compared. Three different models were coded and trained in Pytorch for comparisons.

1. **F2VAE:** The default F2VAE model with a cross-entropy loss and binary classifier for the sensitive label country. The classifier contains 16 length outputs for 16 country classes. Figure 4.6 represents the model architecture.

```
MultiVAE(
  (encoder): Encoder(
    (dropout): Dropout(p=0.5, inplace=False)
    (linear_1): Linear(in_features=632665, out_features=2000, bias=True)
    (linear_2): Linear(in_features=2000, out_features=400, bias=True)
    (tanh): Tanh()
  )
  (decoder): Decoder(
    (linear_1): Linear(in_features=200, out_features=2000, bias=True)
    (linear_2): Linear(in_features=2000, out_features=632665, bias=True)
    (tanh): Tanh()
  )
  (classify): Linear(in_features=200, out_features=16, bias=True)
)
```

Figure 4.6: F2VAE architecture

2. **Multi-Sensitive Attribute VAE (Country):** Our proposed implementation with the new loss function as presented in equation 3.8 and a multi-label classifier for multiple sensitive attributes but still using only one sensitive attribute, country. The same 16 classes as the F2VAE model but a deeper network and different loss function.

```
MultiVAE(
  (encoder): Encoder(
    (dropout): Dropout(p=0.5, inplace=False)
    (linear_1): Linear(in_features=632663, out_features=2000, bias=True)
    (linear_2): Linear(in_features=2000, out_features=400, bias=True)
    (tanh): Tanh()
  )
  (decoder): Decoder(
    (linear_1): Linear(in_features=200, out_features=2000, bias=True)
    (linear_2): Linear(in_features=2000, out_features=632663, bias=True)
    (tanh): Tanh()
  )
  (multi_label_classifier): MultiLabelClassifier(
    (fc1): Linear(in_features=200, out_features=128, bias=True)
    (relu1): ReLU()
    (fc2): Linear(in_features=128, out_features=64, bias=True)
    (relu2): ReLU()
    (fc3): Linear(in_features=64, out_features=16, bias=True)
    (sigmoid): Sigmoid()
  )
)
```

Figure 4.7: Multi-label Classifier implementation (country as sensitive attr)

3. **Multi-Sensitive Attribute VAE (Country, Sex):** Our proposed implementation with two sensitive attributes, country and sex. Classifier output increases to 19 classes, 16 for the country and 3 for sex. At any given time, a user can be attributed to a country and sex.

```
MultiVAE(
  (encoder): Encoder(
    (dropout): Dropout(p=0.5, inplace=False)
    (linear_1): Linear(in_features=632663, out_features=2000, bias=True)
    (linear_2): Linear(in_features=2000, out_features=400, bias=True)
    (tanh): Tanh()
  )
  (decoder): Decoder(
    (linear_1): Linear(in_features=200, out_features=2000, bias=True)
    (linear_2): Linear(in_features=2000, out_features=632663, bias=True)
    (tanh): Tanh()
  )
  (multi_label_classifier): MultiLabelClassifier(
    (fc1): Linear(in_features=200, out_features=128, bias=True)
    (relu1): ReLU()
    (fc2): Linear(in_features=128, out_features=64, bias=True)
    (relu2): ReLU()
    (fc3): Linear(in_features=64, out_features=19, bias=True)
    (sigmoid): Sigmoid()
  )
)
```

Figure 4.8: Multi-label Classifier implementation (Country and sex as sensitive attr)

All the models were trained for 100 epochs with a training batch size of 500. A batch size of 200 was used for testing/validation. The learning rate started at 1e-4 and decreased by 0.1 in epochs 50 and 75. KL divergence factor was introduced during the training with an annealing factor gradually.

4.3.1 Results

This section discusses the results obtained after training the three models. All the hyperparameters were left unchanged as obtained from the default F2VAE model (Stefanidis & Borges, 2022) to only record how the results change by adding our proposed changes.

UFAIR@k (country): The figure below states unfairness results for the models with the users being grouped by their sensitive attribute, country. The best results are in bold and the worst are underlined.

	F2VAE	**Multi-Attr (Country)**	**Multi-Attr (Country, Sex)**
UFAIR@1	<u>0.0149</u>	0.00960	**0.00955**
UFAIR@10	<u>0.0140</u>	0.0126	**0.0125**
UFAIR@20	<u>0.0140</u>	0.0121	**0.0119**

Figure 4.9: UFAIR@K results for Users grouped by country

UFAIR@k (country, sex): The figure below states unfairness results for the models with the users being grouped by two sensitive attributes, country, and sex. The best results are in bold and the worst are underlined.

	F2VAE	**Multi-Attr (Country)**	**Multi-Attr (Country, Sex)**
UFAIR@1	<u>0.0170</u>	0.0040	**0.0020**
UFAIR@10	0.0122	<u>0.0131</u>	**0.010**
UFAIR@20	<u>0.0123</u>	0.0116	**0.0114**

Figure 4.10: UFAIR@K results for Users grouped by country and sex

We can see a decrease in unfairness anywhere from 12% to 30% based on the number of top k values chosen to calculate unfairness between the default model and our implementations. It is to be considered which model is being used to evaluate the results since one of our implementations uses two sensitive attributes to mitigate unfairness.

Our model trained on both a single sensitive attribute and multiple attributes outperform the original F2VAE implementations due multiple reasons. The most significant one being the new deeper architecture of our adversarial classifier which helps the VAE better mitigate the effects of attributes on the recommendations. Secondly, the ability of the classifier to handle multiple sensitive attributes also gives it an edge in mitigating further bias caused by other attributes which is visible in the results for our model trained on multiple attributes. When the recommendations are grouped by country and sex, both our implementations trained on a single and multiple attribute perform well even though one is trained to only mitigate the effects of country, this can be due to the variance in the data presented. The variance in the number of users grouped by country is significantly higher than the number of users grouped by both country and sex. This tells us that when mitigating the effects of the sensitive attribute country, it already covers a high range of the unfairness in the data. Subsequently, our classifier being better able to help the adversarial loss performed good even when trained on a single attribute. Adding the second attribute further helps mitigate unfairness but the label country had more weight as compared to sex. The variance in grouping can be referenced through Figure 4.11.

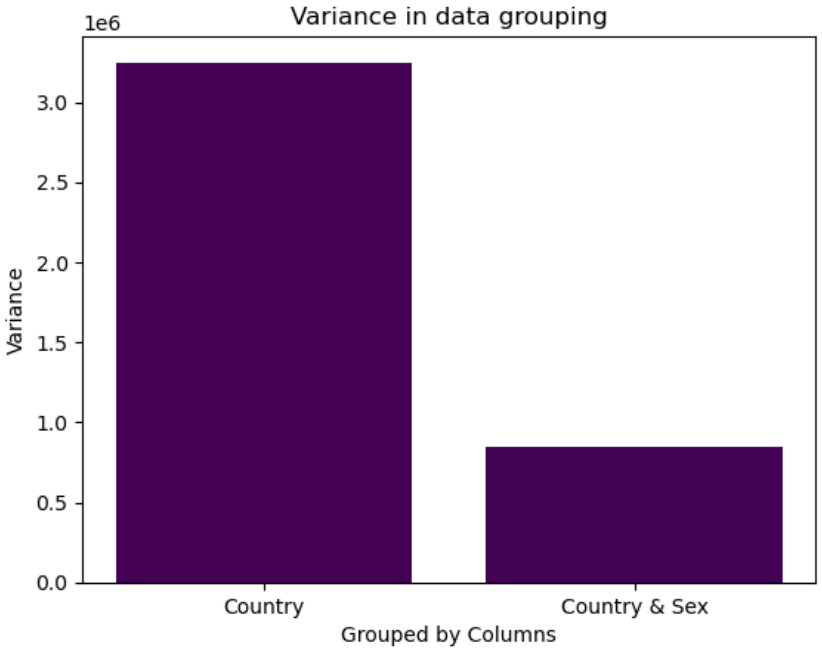

Figure 4.11: Variance in data grouping

The effect of this variance in the data grouping can also be noticed in the results below, where both our implementations outperform the original F2VAE results with the addition of another sensitive attribute providing marginally better results than our implementation trained with only one sensitive attribute.

PREC@k (country): The figure below states the Precision@k values for three different k values, 1, 10, and 20 for users grouped by their sensitive attribute, country. The best results are in bold and the worst are underlined.

	F2VAE	**Multi-Attr (Country)**	**Multi-Attr (Country, Sex)**
PREC@1	0.453	0.465	**0.483**
PREC@10	0.420	**0.438**	**0.438**
PREC@20	0.397	0.412	**0.414**

Figure 4.12: PREC@K results for Users grouped by Country

PREC@k (country, sex): The figure below states the Precision@k values for three different k values, 1, 10, and 20 for users grouped by their sensitive attribute, country, and sex. The best results are in bold and the worst are underlined.

	F2VAE	**Multi-Attr (Country)**	**Multi-Attr (Country, Sex)**
PREC@1	0.453	0.454	**0.470**
PREC@10	0.419	0.430	**0.435**
PREC@20	0.395	0.403	**0.410**

Figure 4.13: PREC@K results for Users grouped by country and sex

We see anywhere from around 4% to 6% increase in overall precision across different comparison parameters. Our implementation with a higher number of sensitive attributes added for mitigation outperforms the default implementation and our implementation with a single sensitive attribute.

NDCG@100: We use the NCDG@100 evaluation metric to examine the ranking quality of our recommenders. We set *k* to 100 to have a better widespread idea of our ranking quality. We have noticed improvement in fairness and the accuracy of our recommender system but it does not always mean a quality improvement. A fairness gain can also sometimes affect ranking quality negatively.

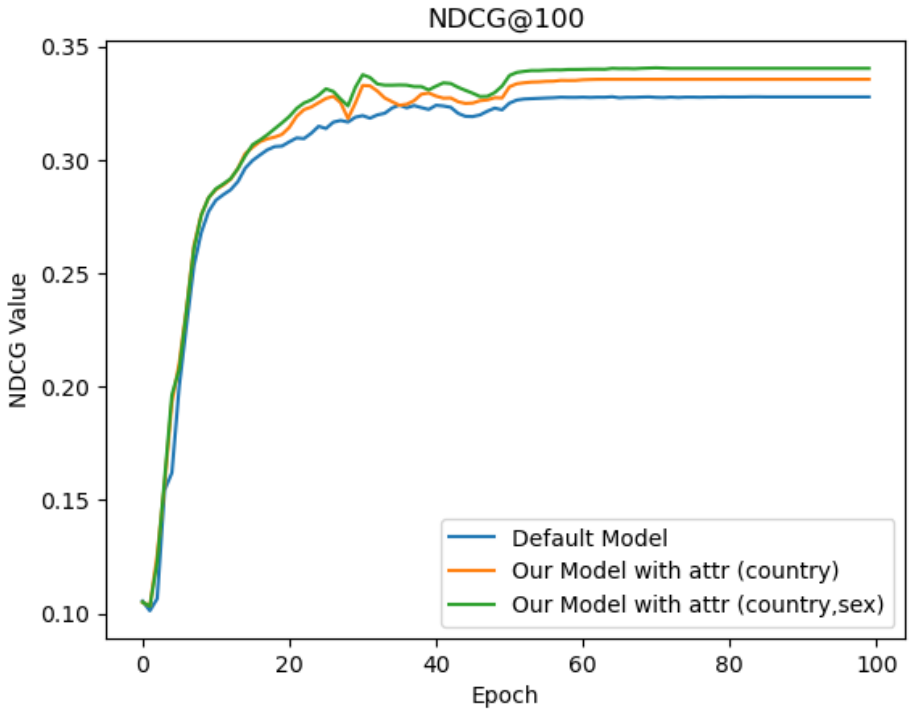

Figure 4.14: NCDG@100 comparison for all models

We evaluate the metric at every epoch during the testing phase after the training phase ends. The improvement trend continues here as well and our implementation provides a 2%-3% quality increase in recommendations alongside reduced unfairness. The Exact figures for NCDG@100 can be referred to in the figure below. The best results are in bold and the worst are underlined.

	F2VAE	**Multi-Attr (Country)**	**Multi-Attr (Country, Sex)**
NDCG@100	<u>0.327</u>	0.335	**0.340**

Figure 4.15: NDCG@100 results for all models.

Recall@100: By focusing on the top 100 recommendations, this metric provides a practical evaluation of the system's effectiveness in capturing a significant number of pertinent items within a reasonably sized recommendation list. This approach is particularly relevant in scenarios where users are confronted with an extensive pool of potential items, as it emphasizes the system's capability to prioritize and present relevant suggestions within a practically manageable subset. As such, the recall@100 metric offers a concise and informative assessment of our recommender system's performance in delivering meaningful recommendations within a larger recommendation space.

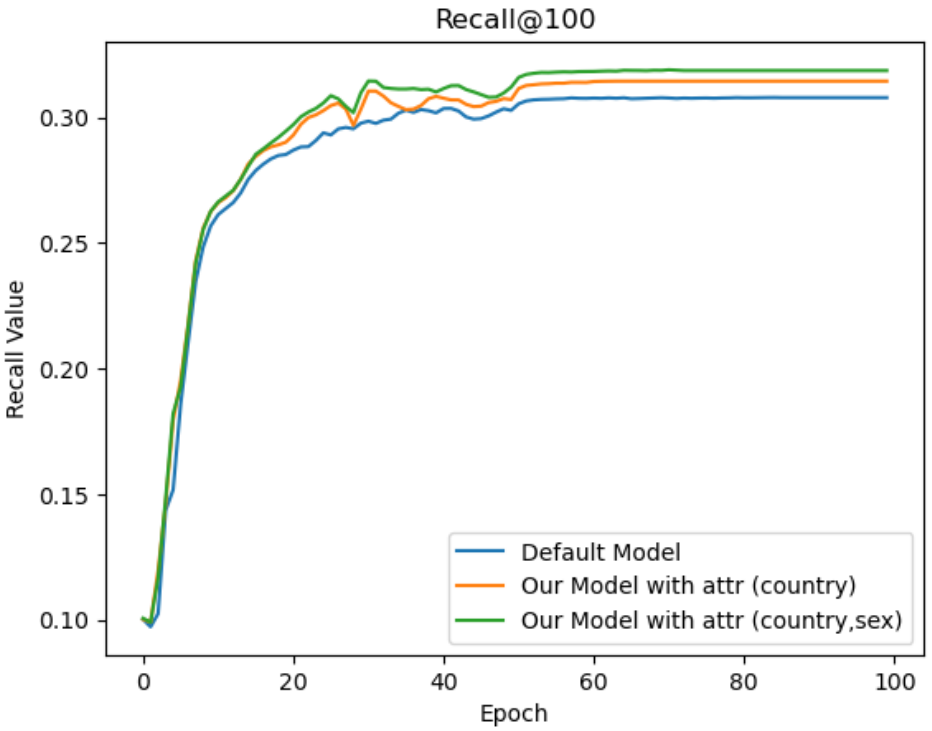

Figure 4.16: Recall@100 comparison for all models.

We get a similar graph to our previous ndcg@100 calculations and our implementations outperform the existing one by giving a 2.5% to 4% performance gain. The exact figures can be referred to in the table below.

	F2VAE	**Multi-Attr (Country)**	**Multi-Attr (Country, Sex)**
Recall@100	0.307	0.314	**0.320**

Figure 4.17: NDCG@100 results for all models.

Classifier loss: Since we used a different classifier type to mitigate sensitive information from the latent vector *z*. The behavior of classifier loss also changed and can be observed in Figure 4.18.

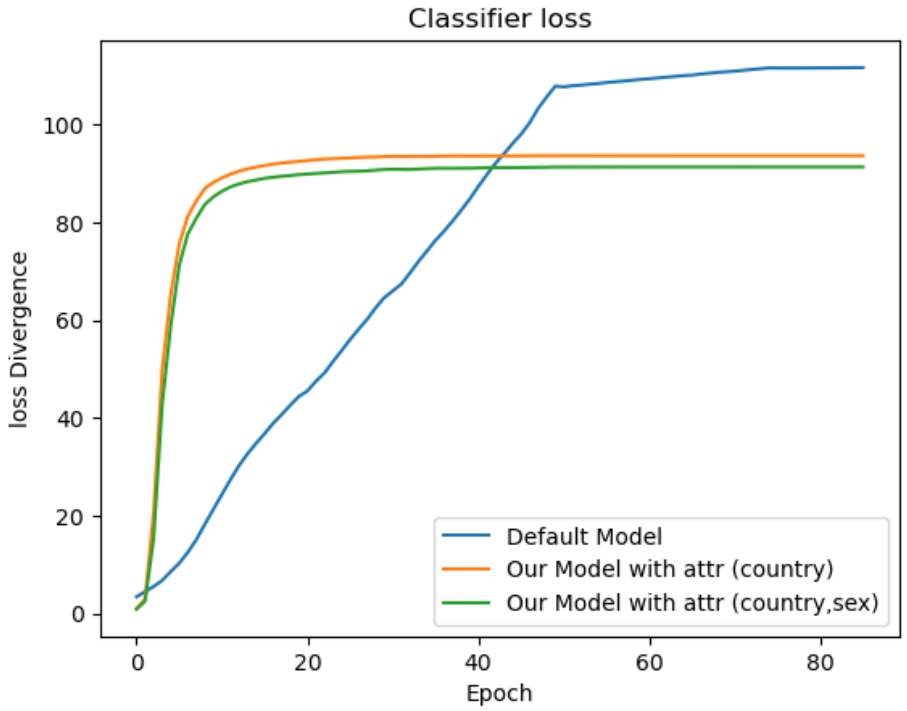

Figure 4.18: Classifier loss across models

It can be observed that our multi-label implementation converges faster than the default implementations which can be attributed to multiple reasons. One is the fact that the new classifier is deeper than the default implementation. The other reason for early convergence of the adversarial loss for the classifier can suggest that the VAE quickly adapts its latent variable representation to better reduce the influence of these sensitive attributes. The model may more effectively find a compromise in the latent space that minimizes the ability of the classifier to predict these attributes.

5. Conclusion

We evaluated the impact of adding a deeper multi-label classifier to mitigate bias and unfairness using multiple sensitive attributes from the dataset as opposed to one. The results obtained conclude an increase in fairness and ranking quality according to all the metrics being evaluated.

By using a deeper classifier tuned to predict multiple sensitive attributes, the model is exposed to more information about the users. The additional information helps the model capture more nuanced patterns and preferences leading to better recommendations. We noticed the largest amount of improvement when training the classifier model with country and sex labels as there were significant variations in the users belonging to a sex within a given country. Predicting a wider range of sensitive attributes also acts as a form of regularization which helps prevent overfitting and improves generalization.

We used a deeper type of network for the classifier and we saw a faster convergence of the loss in that case. The faster convergence in the multi-label case may suggest that the joint optimization is leading to a more effective balance between these competing objectives.

It was also observed that using a deeper classifier increases the computational complexity of the network and adding more sensitive attributes could lead to longer training times and more resources being used.

In future work, we would like to investigate using more sensitive attributes than just two to capture their effects on the model training and results, or, a single new sensitive attribute could be used to capture its results other than using just the country. A new dataset could be tried out which comes with a wider array of demographic information about the users.

References

1. Beel, J. (2016). Research-Paper Recommender Systems: A Literature Survey. *International Journal on Digital Libraries*, *17*(4), 305-338. https://dl.acm.org/doi/10.1007/s00799-015-0156-0
2. Blei, D. M., McAuliffe, J. D., & Kucukelbir, A. (2016, January 4). *Variational Inference: A Review for Statisticians*. arXiv. Retrieved December 13, 2023, from https://arxiv.org/abs/1601.00670
3. Borges, R., & Stefanidis, K. (2019). *Enhancing Long Term Fairness in Recommendations with Variational Autoencoders*. Enhancing Long Term Fairness in Recommendations with Variational Autoencoders. Retrieved December 17, 2023, from https://homepages.tuni.fi/konstantinos.stefanidis/docs/medes19.pdf
4. Borges, R., & Stefanidis, K. (2021, April 22). *On mitigating popularity bias in recommendations via variational autoencoders*. https://dl.acm.org/doi/10.1145/3412841.3442123
5. Borges, R., & Stefanidis, K. (2022, February 22). *Feature-blind fairness in collaborative filtering recommender systems*. https://link.springer.com/article/10.1007/s10115-022-01656-x
6. Breese, J. S., Heckerman, D., & Kadie, C. (2013, January 30). *Empirical Analysis of Predictive Algorithms for Collaborative Filtering*. arXiv. Retrieved December 19, 2023, from https://arxiv.org/abs/1301.7363
7. Burke, R. (2002). *Hybrid Recommender Systems: Survey and Experiments*. SpringerLink. Retrieved December 8, 2023, from https://link.springer.com/article/10.1023/A:1021240730564
8. Çano, E., & Morisio, M. (2019, January 12). *Hybrid Recommender Systems: A Systematic Literature Review*. arXiv. Retrieved November 22, 2023, from https://arxiv.org/abs/1901.03888
9. Chouldechova, A., & Roth, A. (2020). *A snapshot of the frontiers of fairness in machine learning*. https://dl.acm.org/doi/10.1145/3376898
10. Edwards, H., & Storkey, A. (2015, November 18). *Censoring Representations with an Adversary*. arXiv. Retrieved December 15, 2023, from https://arxiv.org/abs/1511.05897

11. Ekstrand, M. D. (2011, May 5). *Collaborative Filtering Recommender Systems*. Now publishers. Retrieved December 2, 2023, from https://www.nowpublishers.com/article/Details/HCI-009
12. Goldberg, D., Nichols, D., & Oki, B. M. (1992, December 1). *Using collaborative filtering to weave an information tapestry*. Using collaborative filtering to weave an information tapestry. https://dl.acm.org/doi/10.1145/138859.138867
13. Gupta, K., Raghuprasad, M. Y., & Kumar, P. (2018, July 14). *A Hybrid Variational Autoencoder for Collaborative Filtering*. arXiv. Retrieved December 20, 2023, from https://arxiv.org/abs/1808.01006
14. Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (n.d.). *Evaluating Collaborative Filtering Recommender Systems*. https://grouplens.org/site-content/uploads/evaluating-TOIS-20041.pdf
15. Hulbatte, S., Wabale, A., Patil, S., & Sathe, N. (2018, October). *Enhanced Job Recommendation System*.
16. Kingma, D. P., & Welling, M. (2013, December 20). *Auto-Encoding Variational Bayes*. arXiv. Retrieved December 13, 2023, from https://arxiv.org/abs/1312.6114
17. Koren, Y., Bell, R., & Volinsky, C. (2009, August 7). *Matrix Factorization Techniques for Recommender Systems*. https://ieeexplore.ieee.org/abstract/document/5197422
18. Leonhardt, J., Anand, A., & Khosla, M. (2018, July 17). *User Fairness in Recommender Systems*. arXiv. Retrieved December 8, 2023, from https://arxiv.org/abs/1807.06349
19. Li, X., & She, J. (2017, August). Collaborative Variational Autoencoder for Recommender Systems. *KDD '17: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 305–314.
20. LI, Y., CHEN, H., XU, S., GE, Y., TAN, J., LIU, S., & ZHANG, Y. (2022, May 26). *Fairness in Recommendation: Foundations, Methods and Applications*. arXiv. Retrieved December 8, 2023, from https://arxiv.org/abs/2205.13619
21. Lika, B., Kolomvastos, K., & Hadjeifthymiades, S. (n.d.). *Facing the cold start problem in recommender systems*.
22. Maćkiewicz, A., & Ratajczak, W. (1993). Principal components analysis (PCA). *ScienceDirect*, *19*(3), 303-342. 10.1016

23. Mcculloch, W. S., & Pitts, W. (1943, December). *A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY*. https://link.springer.com/chapter/10.1007/978-3-642-70911-1_14
24. Melchiorre, A. B., Rekabsaz, N., Parada-Cabaleiro, E., Brandl, S., Lesota, O., & Schedl, M. (2021, September). Investigating gender fairness of recommendation algorithms in the music domain. *ScienceDirect*, *58*(5). https://doi.org/10.1016/j.ipm.2021.102666
25. Pazzani, M. J., & Billsus, D. (n.d.). *Content-Based Recommendation Systems*. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=34446adc7d701a2c3a89c2fc5f6d3479eef407b0
26. Pitoura, E., Stefanidis, K., & Koutrika, G. (2021). Fairness in Rankings and Recommendations: An Overview. Retrieved December 8, 2023, from https://homepages.tuni.fi/konstantinos.stefanidis/docs/VLDBJ2021.pdf
27. Polatidis, N. (2013, October). Recommender Systems: The Importance of Personalization in E-Business Environments. *International Journal of E-Entrepreneurship and Innovation (IJEEI)*, *4*(4), 0-15.
28. Rashevsky, N. (2007, April). The mathematical biophysics of Nicolas Rashevsky. *ScienceDirect*, *88*(3), 178-184. https://www.sciencedirect.com/science/article/abs/pii/S0303264706002140
29. Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. 10.1037
30. Roy, D., & Dutta, M. (2022, May 03). A systematic review and research perspective on recommender systems. *Journal of Big Data*, *9*(59). https://doi.org/10.1186/s40537-022-00592-5
31. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986, 10 9). Learning representations by back-propagating errors. *Nature*, *323*(9), 533-536. https://www.nature.com/articles/323533a0
32. Rumelhart, D. E., & McClelland, J. L. (1987). *Learning Internal Representations by Error Propagation*. https://ieeexplore.ieee.org/document/6302929

33. Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2011). *Item-Based Collaborative Filtering Recommendation Algorithms*. Item-Based Collaborative Filtering Recommendation Algorithms. https://dl.acm.org/doi/10.1145/371920.372071
34. Sedhain, S., Menon, A. K., Sanner, S., & Xie, L. (2015, May). *AutoRec: Autoencoders Meet Collaborative Filtering*. AutoRec: Autoencoders Meet Collaborative Filtering. https://users.cecs.anu.edu.au/~akmenon/papers/autorec/autorec-paper.pdf
35. Shi, X., Ye, H., & Gong, S. (2008). *A Personalized Recommender Integrating Item-Based and User-Based Collaborative Filtering*. https://ieeexplore.ieee.org/abstract/document/5117479
36. Stefanidis, K., & Borges, R. (2022, May 6). *F2VAE: a framework for mitigating user unfairness in recommendation systems*. https://dl.acm.org/doi/abs/10.1145/3477314.3507152
37. Stegemann, J. A., & Buenfeld, N. R. (1999). *A Glossary of Basic Neural Network Terminology for Regression Problems*. https://link.springer.com/article/10.1007/s005210050034
38. Stitini, O., Kaloun, S., & Bencharef, O. (2023). *Towards a robust solution to mitigate all content-based filtering drawbacks within a recommendation system*.
39. Thorat, P. B., Goudar, R. M., & Barve, S. (n.d.). *Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System*. https://d1wqtxts1xzle7.cloudfront.net/59762468/10.1.1.695.642820190617-91457-z4s1rf-libre.pdf?1560756140=&response-content-disposition=inline%3B+filename%3DSurvey_on_Collaborative_Filtering_Conten.pdf&Expires=1701907979&Signature=cjXfF1w8jPjHnATdAhm63DCkKi
40. Wang, Y., Ma, W., Zhang, M., Liu, Y., & Ma, S. (2023, February 7). *A Survey on the Fairness of Recommender Systems*. YouTube. Retrieved December 8, 2023, from https://dl.acm.org/doi/10.1145/3547333
41. Zhang, R. (2014). Collaborative Filtering for Recommender Systems. *Second International Conference on Advanced Cloud and Big Data*. https://ieeexplore.ieee.org/document/7176109/figures#figures

42. Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019, July 10). *Deep Learning based Recommender System: A Survey and New Perspectives*. arXiv. Retrieved December 19, 2023, from https://arxiv.org/abs/1707.07435

43. Zhu, Z., He, Y., Zhao, X., Zhang, Y., Wang, J., & Caverlee, J. (2021, March 8). *Popularity-Opportunity Bias in Collaborative Filtering*. Popularity-Opportunity Bias in Collaborative Filtering. https://dl.acm.org/doi/10.1145/3437963.3441820

44. Zisopoulos, C. (n.d.). *(PDF) Content-Based Recommendation Systems*. ResearchGate. Retrieved November 22, 2008, from https://www.researchgate.net/publication/236895069_Content-Based_Recommendation_Systems