Building Automatic Agendas for AAC using a recommendation system with deep learning models

Advanced Computer Architectures Project report

Angela Remolina
Politecnico di Milano
angelasofia.remolina@mail.polimi.it

Sergio Pardo
Politecnico di Milano
sergio.pardo@mail.polimi.it

Abstract

This paper explores the development of an automated agenda creation system for Augmentative and Alternative Communication (AAC) using a recommendation system powered by deep learning models. The primary objective is to assist educators in designing personalized schedules for individuals with communication disorders. Utilizing TensorFlow Recommenders, the system employs content-based filtering techniques, allowing educators to input external information and create customized activity schedules tailored to the needs of their students. The recommendation system adapts dynamically through reinforcement learning, continuously refining its suggestions based on user interactions and feedback. This approach not only aims to enhance the effectiveness of AAC tools but also to promote inclusivity by addressing the diverse needs of individuals with communication impairments. The proposed model demonstrates significant potential facilitating better educational tools and developmental outcomes.

1. Introduction

Inclusion is a constant topic debated in today's society and represents a challenge in many ways, specially, when it comes to removing the barriers that usually set apart disabled people from our society. No matter the cause, disease, injury or developmental disorders, it is of the utmost importance to keep innovating and trying new ways of overcoming the marginalization of this part of the population. There are many types of disabilities, but this article will mainly focus on the ones related to communication.

A communication disorder is an impairment of any of the processes by means of which speakers produce, and hearers comprehend, spoken, written, or signed utterances [2]. Given this definition, it is possible to see that there is not a single way in which this condition express itself, aggravated by the fact that the age, socioeconomic, ethnic and racial backgrounds of individuals in this position are widely ranged [3]. In other words, a solution that perfectly works for every person with apparently the same problem does not exist. This is where modern computer science can play a significant role by applying a series of models that serve multiple purposes and can be tailored for tasks and users alike.

This paper takes the AMBRA (*Pervasive and Personalized Augmentative and Alternative Communication based on Federated Learning and Generative AI*), approach [6] as a basis to keep on exploring the possibility of having an open contribution platform where professionals, working on improving on the tools and living conditions for people with communication disorders, will be able to share their knowledge and the contents created with other educators. The key point to this platform, is that it will integrate several technologies, getting the best publicly available methods of state-of-the-art models for each task, enhancing the process of content creation and the possible adaptability of each solution to a particular case. The AMBRA platform takes as foundation the Augmentative and Alternative Comunication (AAC), or better, it is a new proposed implementation of it.

Given this background, the creation of agendas are a key point to educators as they play an important role in the life of the students with disabilities. They can help decrease the feeling of anxiety, serve as a tool to develop desired skills on a daily basis and ease the construction of a bedtime routine that is critical for the quality of life [1].

This last element in the context of communication disabilities is precisely the scope of this project. Particularly, the creation of visual and customised schedules using

the latest deep learning models available in the cloud. The objective is facilitating this process to the educators, making them suggestions of activities that could be useful to a particular individual based on its characteristics and the activities that have been already chosen for them.

2. Theoretical framework

Before the beginning of the study it is needed to clarify each concept that is relevant to the project in matter. Some of these terms and definitions are enumerated in this section.

2.1. Recommender Systems

Recommender systems (RS) have become integral to many digital platforms, assisting users by suggesting products, services, or content tailored to their preferences. These systems operate by analyzing user behavior and leveraging data to provide personalized recommendations. Broadly, recommender systems can be categorized into three main types [5]:

1. Content-Based Filtering: This approach consists in creating profiles for users and/or products to extract significant qualities that could shine some light on the user's preferences. These profiles are then used to try to find the best match between users and products. External information is often required by this method to be able to build the user's profile, so it is not always an easy method to implement.

2. Collaborative Filtering: This method predicts a user's preferences based on the preferences of other users. It creates an implicit profile of the user based on previous experiences with other products. A drawback to collaborative filtering is it suffers from *cold start*; the system may have trouble addressing new products or users to the environment since there is no historical information.

   The two major sub-areas to collaborative filtering are:

   Neighborhood Collaborative Filtering:

   Consists in finding users and items that are close or similar to each other. This means there will be neighborhoods of users and items and they will be related to each other depending on the implementation. Algorithms like k-nearest neighbors (k-NN) are often used in this context.

   Latent Factor Collaborative Filtering:

   This method aims do discover the dimensions that best describe products and users. You could discover natural dimensions, for example, the genre of a movie, but also some hidden dimensions like the pace of said movie. These dimensions can be compared between products to discover similar products and the users that like them.

3. Hybrid Methods: These combine content-based and collaborative filtering techniques to overcome the limitations of each approach. By integrating multiple data sources and recommendation techniques, hybrid methods aim to provide more accurate and diverse recommendations.

2.2. Reinforcement Learning

Reinforcement learning (RL) is a type of machine learning where an agent learns to make decisions by performing actions and receiving feedback from its environment. Unlike supervised learning, which requires labeled input-output pairs, RL focuses on learning from the consequences of actions. Key components of reinforcement learning include:

- Agent: The decision-maker that interacts with the environment.
- Environment: The external system the agent interacts with.
- State: A representation of the current situation of the environment.
- Action: Choices made by the agent that affect the state.
- Reward: Feedback from the environment based on the action taken.
- Policy: The strategy that the agent employs to determine its actions.
- Value Function: Estimates the expected long-term return of states or actions, guiding the agent to make optimal decisions.

3. Integration of Recommender Systems and Reinforcement Learning for Personalized Agendas

Combining recommender systems with reinforcement learning can significantly enhance the personalization of agendas. Here's how this integration works[4]:

1. Dynamic Adaptation: Reinforcement learning enables recommender systems to adapt dynamically to user feedback. By continuously learning from user interactions, the system can adjust recommendations in real-time, offering a more personalized experience.

2. Exploration and Exploitation: Reinforcement learning balances exploration (trying new recommendations) and exploitation (leveraging known user preferences). This balance ensures that the system not only caters to current user preferences but also discovers new interests.

3. Context-Awareness: With reinforcement learning, recommender systems can incorporate contextual information, such as time of day, location, and user activity patterns, to provide more relevant suggestions.

4. Long-Term Optimization: Reinforcement learning focuses on long-term rewards, helping to optimize user engagement over time rather than just immediate satisfaction. This approach can improve user retention and satisfaction by aligning recommendations with the user's evolving needs and preferences.

These 2 fields of study are of big interest to tech giants, so they keep being researched on and probably new models will come in the next years.

Personalized agendas benefit from this integration by providing users with tailored schedules and reminders based on their habits, preferences, and goals. For instance, a personalized agenda system can recommend activities that align with the individual's patterns of behaviour, preferred activities and the activities that help them the most throughout the process. By employing reinforcement learning, the system can refine its recommendations through continuous interaction, ensuring that the agenda evolves with the user.

This is the potential that current state-of-the-art models and technology have in helping educators in their mission to make this society a more inclusive one. Although the present implementation does not achieve this extend, it is possible to see through this review that AAC can be enhanced by taking advantage of today's technology and ingenuity.

4. Proposed approach

The approach taken is to build a system capable of personalizing agendas based on the context (fixed activities) and the preferences of the user (the educator planning the schedule of his pupil) was based on recommender systems. More specifically it was decided the system would be a content-based filtering, given the fact that educators know the profile of the individual to which they are designing the schedule and have external information they could input into the system, in order to trace to them the most suitable activities. Nonetheless, the most prioritized attribute of the RSs was the learning one; the system will learn on each individual preference's and will try to extrapolate this learning to other individuals with the same characteristics. Hence, the system would be able to provide recommended activities for the schedules of individuals who have never had this type of attention before, boosting the progress of their learning. To illustrate the interaction of a user with the proposed system, a flow diagram is shown below 1 2.

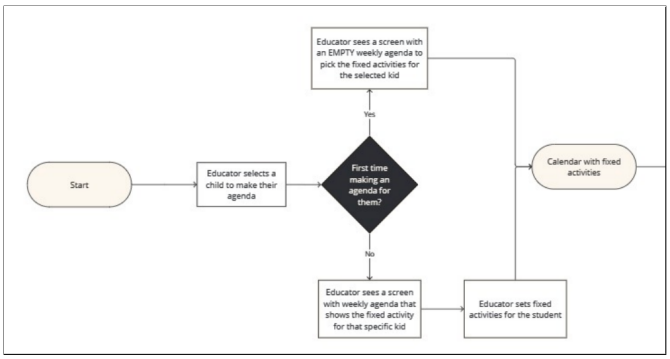

Figure 1. User's flow diagram part 1

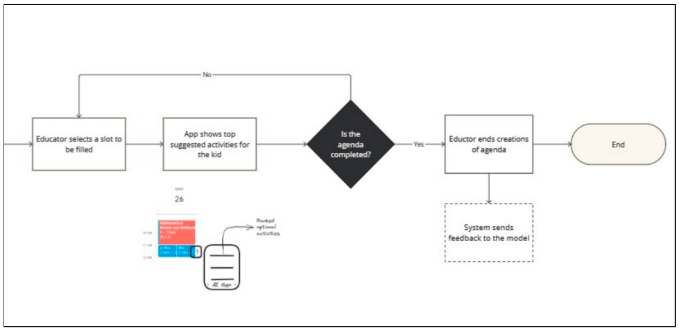

Figure 2. User's flow diagram part 2

5. System architecture and implementation

5.1. General system architecture

The complete system architecture consists of three main components that interact with each other. The workflow can be seen below in figure 3.

1. **PostgreSQL Database**: This database stores all the activities and user data. It is accessed via an API that supports CRUD operations (Create, Read, Update, Delete).

2. **Python Model with TensorFlow**: This model is responsible for generating activity recommendations. It has its own API to receive recommendation requests and user feedback.

3. **General API in Node.js**: This API serves as the intermediary, connecting user queries to the database and

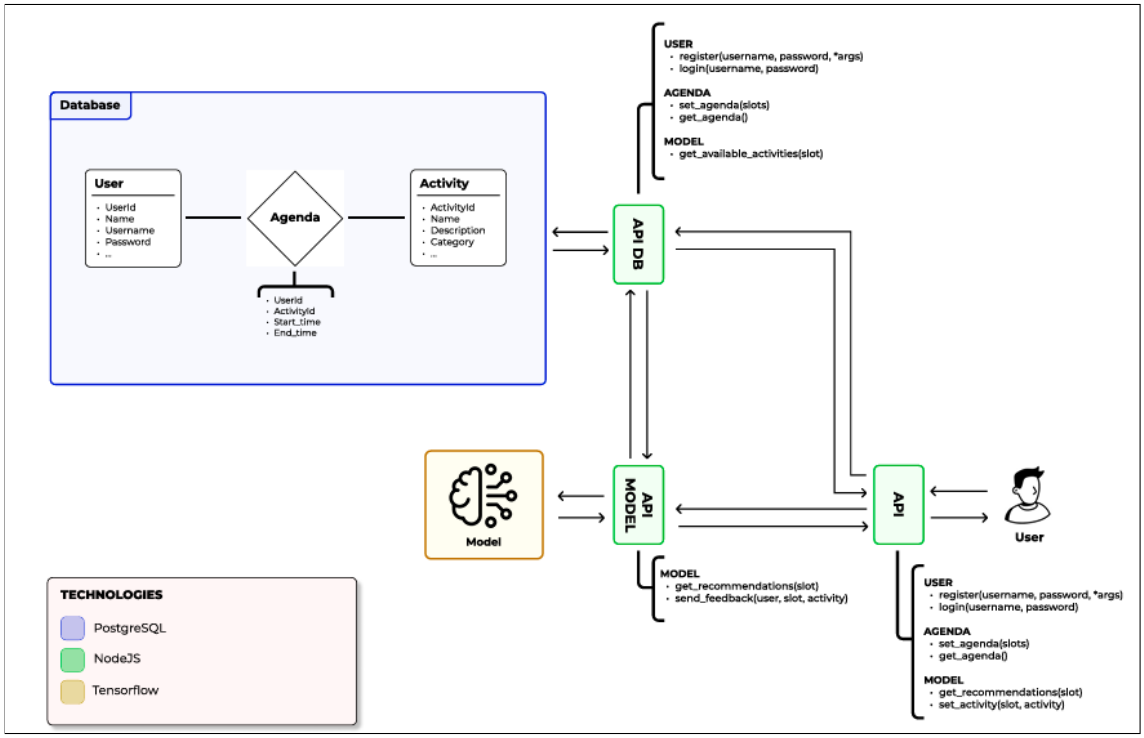

Figure 3. Proposed architecture for the system.

the recommendation model. It handles the communication between the front-end (user requests) and the other components

5.2. The workflow

1. User Query: The user makes a query through the general API in Node.js.
2. Database Interaction: The general API queries the PostgreSQL database to retrieve necessary information about activities and users. E.g. Request to show the agenda for that specific user.
3. Recommendation Generation: The general API requests recommendations from the TensorFlow model in Python.
4. Response to User: The general API sends the generated recommendations back to the user.
5. User Feedback: The user provides feedback on the recommendations, selecting the activity that wants to add to his agenda, which is sent to the general API and then to the Python model to improve future recommendations.

This architecture ensures each component specializes in specific tasks and all interactions are efficiently managed through the APIs.

5.3. Recommendation model components

The recommender model is a reinforcement learner, specifically an adaptive learning system. To illustrate its functioning see the diagram below that is useful for understanding how adaptive learning systems personalize content and improve user experience over time.4.

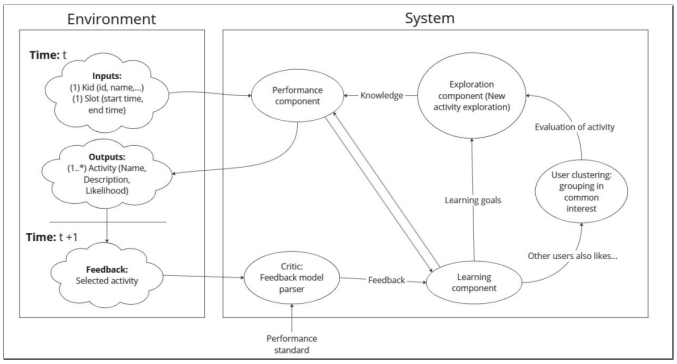

Figure 4. Model components

This diagram is showing interactions between an *Environment* and a *System*.

At time t: Environment inputs the slot and the user's information (exact information explained in the *two-tower retrieval model architecture* subsection). Then it continues to the performance component. This component interacts with the external environment. It receives knowledge or

information from outside sources. Then it reaches the exploration component, that is responsible for exploring new activities not fixing for single choices from before. It likely generates recommendations or suggestions based on user preferences. Also the performance component sends and receives information from de Learning Goals Component, that represents the system's learning objectives or targets. Finally, the performance component gives an output with the top recommended activities for the user in matter.

At time t+1: The user selects the activity it wishes from the recommended activities list obtained at the output. And then this feedback is sent back to the system, specifically to the Critic Component. This component processes feedback, where it is likely to evaluates user interactions, performance, and learning goals. The feedback loop influences both learning goals and performance standards.

5.4. Two-tower retrieval model architecture

Selecting the architecture of the model is a crucial step in the modeling process of the recommendation system. Since there are two entities the model should have into account: **User** and **Activity**, then a two-tower retrieval model is the chosen for the model architecture. Each tower is constructed independently before combining them into the final model. The building is illustrated in the following figure 5.

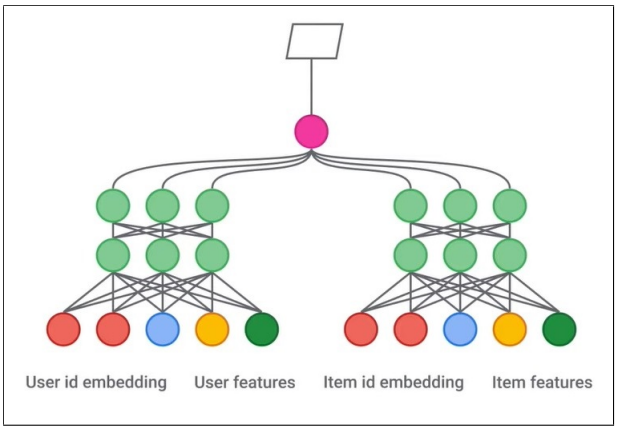

Figure 5. Tower model. Image taken from [7]

In the context of this project the *item* named above refers to an activity to be recommended e.g. Do yoga, play chess, etc. So *"item id embedding"* refers to the **Activity ID** embedded into a tensor and *"item features"* refers to **Activity features**. The features taken by the model are the name of the activity and its category (sport, boardgame, dancing, etc.). The *"User id emmbeddings"* and *"User features"* represent the corresponding user information and characteristics.

To model the desired environment, 2 main entities were modeled: the user, meant as the individual to which the schedule is assigned, although the person designing the schedule is the educator. And the Activities that can be included in the schedule. This simple model allows each user to have a set of activities they will be doing based on the days they are assigned to them.

The activities include the following attributes:

- **id**: unique activity identifier
- **title**: name of the activity
- **category**: An assigned category that adds value to the recommender system's search.
- **description**: Description of the activity itself.
- **start_time**: Date and time at which the activity starts (which can be chosen by the educator)
- **end_time**: Date and time at which the assigned activity will end (coherent with **start_time**)
- **always_open**: indicates if the activity is free of time restrictions.
- **image**: visual representation of the name of the activity.

Whereas the user's attributes are the following:

- **id**: unique user identifier
- **name**: complete name of the user (the person using the agenda).
- **username**: unique username for the user in the system.
- **email**: unique email for the user.
- **hashed_password**: a codified version of the users password to be verified during the login process.

6. Results

6.1. System prototype

A prototype of the proposed system was developed to demonstrate its functionality. The system architecture includes a PostgreSQL database with a CRUD API, a recommendation model built using TensorFlow in Python, and a general API in Node.js that integrates user queries, database interactions, and the recommendation model. Additionally, a simple frontend was created using Handlebars to visualize the system better. It is important to note that this is a low-fidelity prototype, and more sophisticated designs can be implemented to enhance the user experience. See figure 6

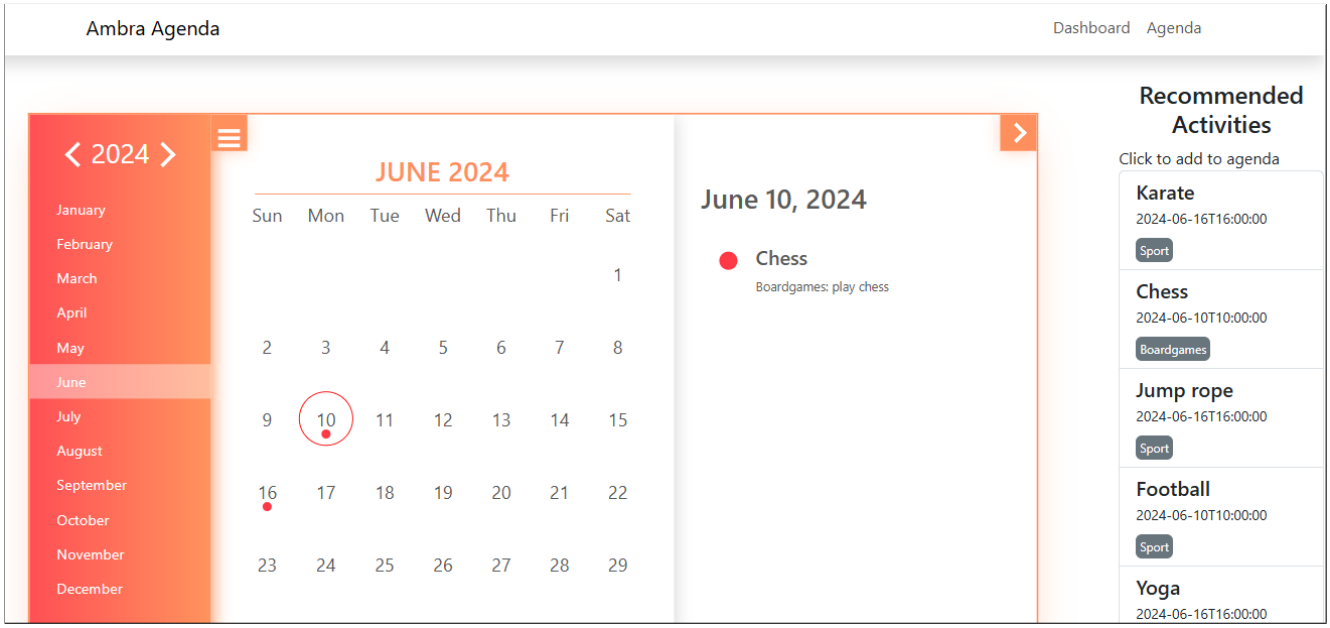

Figure 6. Prototype for the proposed system.

6.2. Recommender System Results Overview

To show how the model gives recommendation, an example with fictional users is provided below 7

User ID	User name	Past selections		Recommendation	
		Activity	Category	Activity	Category
89f5bb79-	Ana	Play UNO	Boardgame	Play UNO	Boardgame
		Salsa	Dancing	Salsa	Dancing
				Jenga	Boardgame
				Yoga	Sport
				Drawing	Leisure
ff4f4dc8-8	Bob	Jump rope	Sport	Karate	Sport
		Karate	Sport	Chess	Boardgame
		Karate	Sport	Jump rope	Sport
		Karate	Sport	Football	Sport
		Chess	Boardgame	Jenga	Boardgame
		Chess	Boardgame		
80292076-	Carlos	Play UNO	Boardgame	Play UNO	Boardgame
				Singing	Leisure
				Yoga	Sport
				Jenga	Boardgame
				Salsa	Dancing

Figure 7. Recommendations for simultated users

The table demonstrates the effectiveness of our recommender system by showcasing the activities it recommends to users based on their past selections. The recommendations align closely with users' interests, ensuring they receive personalized and relevant suggestions. An analysis for each user is provided below:

1. **User: Ana**

   Ana has previously chosen Play UNO and Salsa, indicating a preference for boardgames and dancing activities. The system accurately recommends similar activities such as Jenga, another boardgame, and Salsa. Additionally, it broadens the scope with related activities like Yoga, which falls under sport, and Drawing, under leisure, both offering Ana a variety while staying within her interest zones.

2. **User: Bob**

   Bob's repeated selection of Karate highlights a strong interest in this sport, and the system recommends it again, acknowledging his preference. His interest in sports is further supported by recommendations like Jump rope and Football. The inclusion of Chess and Jenga, both boardgames, aligns with his past interest in Chess, offering variety within the same category.

3. **User: Carlos**

   Carlos has selected Play UNO previously, showing an inclination towards boardgames. The system smartly recommends Play UNO again, along with another boardgame, Jenga, catering to his known interests. Additionally, it suggests Singing, Yoga, and Salsa, which introduce Carlos to activities from other categories he might enjoy, thus providing a well-rounded set of recommendations.

With these small examples, it is shown that the recommender system effectively personalizes activity suggestions by analyzing users' past selections. For instance:

Ana receives recommendations that reflect her interest in boardgames and dancing, with additional leisure and sport options. Bob's suggestions emphasize his strong preference for sports, especially Karate, while also incorporating boardgames. Carlos is introduced to a broader range of activities while staying true to his interest in boardgames. By doing so, the system ensures users are likely to engage with and enjoy the recommended activities, demonstrating the system's accuracy and reliability in predicting user preferences

7. Conclusions and future work

A glimpse into the world of communication disabilities was taken. Afterwards, some of the possibilities that modern techniques and algorithms offer to AAC were tackled on this paper, focusing personalized schedules via the recommendation of activities for people with communication problems, either permanent or ambulatory. Consequently, a possible implementation with tools available for free on the cloud, particularly tensorflow, was developed to show an initial prototype for a potential system to be. As it was said previously, this system takes as foundation the content-base filtering of Recommender Systems, therefore, the extension of the system with a collaborative filtering part could improve it, specially in finding neighbour users, that tend to exhibit the same patterns. As also said in section 2, the fusion with reinforcement learning can improve the performance of the system in the long run.

A. Supplementary Material

The source code for this project can be found at https://github.com/AngelaRemolina/AutomaticAgenda-AMBRA

References

[1] Bergeron healthcare. Routines and Schedules for Children with Special Needs, 2024. 1

[2] L. Cummings. Communication disorders: A complex population in healthcare. 1(2):88, 2023. 1

[3] J. L. David Beukelman. *Augmentative Alternative Communication: Supporting Children and Adults with Complex Communication Needs*. Paul H Brookes, 2020. 1

[4] A. et al. Reinforcement learning based recommender systems: A survey. 2022. 2

[5] B. . V. Koren. Matrix factorization techniques for recommender systems. *IEEE*, 2009. 2

[6] C. Pilato. Foundation models in augmentative and alternative communication: Opportunities and challenges. page 8, 2024. 1

[7] TensorFlow. Tensorflow recommenders documentation, 2024. https://www.tensorflow.org/recommenders. 5