Group 26 - EB-NeRD RecSys Challenge

MAXIMILIAN RANZINGER, TU Wien, Austria
PIA SCHWARZINGER, TU Wien, Austria
PHILIPP VON DER THANNEN, TU Wien, Austria
YAHYA JABARY, TU Wien, Austria

The 2024 RecSys group project provided a challenge opportunity to gain hands-on experience with common news recommendation systems, using the Ekstra Bladet News Recommendation Dataset (EB-NeRD), which encapsulates user-article interactions for Danish news reports. As part of the course *Recommender Systems* team – Group 26 – participated in this challenge with the aim of implementing effective algorithms to predict user engagement with news articles and tuning their parameters for the best possible outcome achievable with the resources available. The models considered in this process are NRMS (Neural News Recommendation with Multi-Head Self-Attention) as the general baseline, GRU (Gated Recurrent Unit) as model assigned to us and LSTUR (Neural Recommendation with Long Short-Term Memory User Representation) as free choice. In this report, we introduce the the task, particularly focusing on the content and characteristics of the data and the models employed. Additionally, we detail all implementations (including a description of the technical system issues encountered), enhancements, and their respective outcomes. Empirical results are presented and specifically analyzed with respect to differences between the two models based on GRU layers especially. The adaptation of the models with respect to Beyond Accuracy Metrics was not implemented due to the smaller group size of four (agreement with the course organizers).

ACM Reference Format:
Maximilian Ranzinger, Pia Schwarzinger, Philipp von der Thannen, and Yahya Jabary. 2024. Group 26 - EB-NeRD RecSys Challenge. 1, 1 (July 2024), 12 pages.

1 Introduction

The 2024 ACM RecSys Challenge, integrated as a group project within the *Recommender Systems* course, necessitated an in-depth engagement with news recommendation systems. This involvement aimed not only to participate on the internal challenge leaderboard but also to facilitate substantial learning outcomes. Utilizing the Ekstra Bladet News Recommendation Dataset (EB-NeRD), which encapsulates user-article interactions for Danish news reports, it is the challenge's objective to predict user engagement with news articles accurately. This involved implementing advanced recommendation algorithms and fine-tuning their parameters to enhance performance. For all activities on the challenge's Codabench platform our team is operating under the group name `Group 26+`. A mention should be given to the main organizers of the challenge: Ekstra Bladet and JP/Politikens Hus A/S [3]. Our internal workload distribution can be found in section 6.1.

To evaluate the models, the challenge task makes use of several standard metrics in the recommendation field, ensuring a comprehensive assessment of their performance. These metrics included the area under the ROC curve

Authors' Contact Information: Maximilian Ranzinger, TU Wien, Vienna, Austria; Pia Schwarzinger, TU Wien, Vienna, Austria; Philipp von der Thannen, TU Wien, Vienna, Austria; Yahya Jabary, TU Wien, Vienna, Austria.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

(AUC; measurement of the ability of the model to distinguish between clicked and non-clicked articles), the mean reciprocal rank (MRR; evaluating the effectiveness of the recommendations by considering the rank position of the first relevant item) and the normalized discounted cumulative gain (nDCG@K; accounting the relevance and rank position of the top-K recommendations) [3]. Our group used the validation loss internally as important metric for the generalization ability of a machine learning model. Additionally, to address the normative complexities inherent in news recommendations, the challenge's evaluation incorporates samples specifically designed to assess models based on Beyond-Accuracy Objectives such as intra-list diversity, serendipity, novelty, and coverage [6]. After consultation with the course organizers regarding the reduced group size of four, it has been determined that our selected models do not require optimization with respect to these aspects.

According to the requirements of the university exercise, our implementation of the challenge comprises three different models. NRMS (Neural News Recommendation with Multi-Head Self-Attention) serves as the general baseline to be implemented, GRU (Gated Recurrent Unit) as the model assigned to us, and LSTUR (Long Short-Term Memory User Representation) as the group's choice. Our team's selection of the free option was heavily influenced by the basic implementation of LSTUR, utilizing GRU layers within the provided code framework. Consequently, we developed an internal research question to determine how the results of the prediction quality from the simple implementation of GRU compare to those of the LSTUR architecture. All implementations were integrated into the provided repository, *ebnerd-benchmark* [5], with the implementation for GRU being developed independently of the given benchmark with reference to the code collection *recommenders* [9].

The steps carried out for model tuning to enhance the quality of results included preprocessing activities, such as comparing different document embeddings where applicable (*xlm-roberta-base* as well as *bert-base* in English and the articles' language Danish), and particularly hyperparameter tuning. For this purpose, *TensorFlow Keras Tuner* using *Hyperband* [7] was employed, and nearly all attributes of the base model were addressed depending on the model. Examples of these attributes include history size, head number and dimension, learning rate, activation function, and window size.

In this report, we introduce the task focusing on the data's content and characteristics and the models employed. We also detail all implementations, including technical system issues encountered, enhancements made, and their respective outcomes. Finally, we present and analyze empirical results, specifically examining the differences between the two models based on GRU layers: the GRU in its pure implementation and LSTUR.

2 Data

The RecSys 2024 Challenge dataset, referred to as EB-NeRD (Ekstra Bladet News Recommendation Dataset) [2], is an extensive curated collection designed for developing and assessing news recommendation systems. Provided by content of Ekstra Bladet, this dataset showcases the complexities of news recommendation in an authentic environment, capturing elements such as content dynamism, user interest shifts, and the imperative to balance accuracy with diversity.

Overall the data encompasses interactions from over 2.3 million users, with more than 380 million impression logs gathered from Ekstra Bladet's platform over a period of six weeks from April 27 to June 8, 2023 [3]. It includes content from approximately 125,000 news articles, enriched by features including user demographics, article metadata, contextual information such as title, abstract, body, and categories to allow different recommendation strategies. Notably, each article is enhanced with textual context features and proprietary model-generated features such as topics, named entity recognition (NER), and article embeddings. These embeddings facilitate content-based recommendation strategies

by allowing for nuanced semantic comparisons between articles. To ensure privacy, user data was anonymized by securely hashing each user into an anonymized ID using one-time salt mapping [2].

A significant aspect of EB-NeRD is its structured temporal layout, crucial for developing models that account for temporal variations in user preferences. The dataset is segmented into distinct sets: demo, small, large and a dedicated test set, delivered in column-oriented parquet files. Each training bundle contains a training set and a validation set, along with associated articles. Data splits include behavior logs for a 7-day period and users' click histories, representing 21 days of clicked news articles prior to the behavior logs' period. Click histories are fixed and do not update during the data split period.

Challenged participants are allowed to use all provided open-source data, which includes metrics such as Total Inviews and Total Pageviews. The competition's framework promotes the sharing of innovative techniques and results, with a focus on transparent reporting of outcomes using various features, both standard and controversial. A feature of the evaluation process marked as interesting is the inclusion of 200,000 synthetic samples specifically designed to assess metrics beyond accuracy. These samples, uniform in terms of article inviews, impression-ID, and timestamp, enable controlled evaluations of recommendation diversity and novelty. Baselines and scripts to generate these samples are provided, standardizing this crucial aspect of recommendation quality evaluation [2]. As mentioned before, this aspect of recommendation is not addressed in our group work.

3 Approach

In this section, aspects of our practical implementation are described. Initially, the system environment and difficulties encountered that were pertinent to training the models and making predictions are discussed. Subsequently, we detail the models implemented and the process of their implementation. This includes a description of the adaptation of parameters that were adjusted to enhance model performance.

3.1 System Environment

The exercise specifications provided three options regarding the hardware for running the models: execution on local hardware, use of the TU Wien JupyterHub, or utilization of Google Colab. Regarding all options, various issues arose across all groups, which can also be reviewed in the TUWEL forum forum of the course. For the sake of flexibility in basic platform usage, our group focused on attempting to use local hardware and the JupyterHub.

It should be noted that all our developments were exclusively executed on the small instance of the dataset, but the total fraction of 100 % was used. The primary bottleneck for utilizing the large dataset has been shown to be the high demand for RAM, which exceeds the capacity of both local and JupyterHub machines. Additionally, the JupyterHub cannot be directly used with conda environments, which are required for submission and JIT (Just-In-Time) compilation errors occurred when using the GPU. The recommended workaround (see [forum post]), which involves using only the CPU, proved to be impractical in terms of computation time. An attempt to run the models on JupyterHub without the conda environment try proved to be less efficient than the local hardware.

The most suitable solution for all computations was found to be a locally used workstation with GPU support. This setup included 32 GB of DDR5 memory and an NVIDIA RTX 4080 (16 GB). Again, RAM proved to be a limitation, as *TensorBoard* required 42 GB for the small dataset and not used due to that in contrast to the initial framework implementation. Using this configuration, acceptable runtimes in optimization were achieved using the *TensorFlow Keras Tuner* (initially: NRMS around 2 hours; LSTUR: 10 hours).

3.2 Prediction Models

As given by the taks the approach to the challenge should include three distinct models: NRMS (Neural News Recommendation with Multi-Head Self-Attention) as the standard baseline, GRU (Gated Recurrent Unit) as our assigned model, and LSTUR (Neural Recommendation with Long Short-Term Memory User Representation) chosen by our group. Our choice of LSTUR, influenced by its integration of GRU layers as specified in the coding framework, led us to investigate how its prediction quality compares to a straightforward GRU implementation. This was used internally as additional research question.

NRMS [10]. The NRMS model utilizes multi-head self-attention mechanisms to discern semantic relationships within and between news articles and users. The model features an encoder that converts news articles into high-dimensional embeddings, highlighting essential elements like titles and abstracts. Each attention head within the multi-head layer focuses on distinct segments of the input, enabling the model to capture various facets of article content. By synthesizing these insights, NRMS compiles contextual information, offering a refined understanding of user preferences and improving recommendation accuracy. This approach is particularly beneficial for news recommendation systems due to its dynamic processing and evaluation of textual features. The *recommenders* repository categorizes the model as Content-Based Filtering [9].

GRU [4]. The GRU model, a variant of recurrent neural networks (RNN), is optimized for processing sequential data, which is key for modeling temporal user interactions. GRUs overcome the vanishing gradient issue that plagues standard RNNs through innovative gating mechanisms – the reset gate and the update gate – that manage information flow. The reset gate helps the model determine the extent of past information to forget, whereas the update gate controls the integration of new information. This structure makes GRUs adept at capturing temporal patterns and dependencies, which is crucial for predicting user behaviors based on their reading histories in news recommendation scenarios. The *recommenders* repository categorizes its plain GRU implementation for sequential data as Collaborative Filtering [9].

LSTUR [1]. The LSTUR model incorporates both long-term and short-term user preferences through a hybrid architecture that combines user embeddings with GRU layers. The long-term preferences are encapsulated by static user embeddings, derived from users' reading histories, reflecting their enduring interests. In contrast, short-term preferences are modeled using GRU layers that capture recent user behaviors, indicative of current interests. This combination allows LSTUR to adjust to changes in user preferences over time while maintaining a consistent grasp of their fundamental interests, thereby delivering personalized recommendations that align with both recent activities and established preferences. The *recommenders* repository categorizes LSTUR model as Content-Based Filtering [9].

3.3 Implementation

The implementations of NRMS and LSTUR were essentially available in the *ebnerd-benchmark* [5] repository and needed to be transformed into a representative format. Additionally, several code modifications were made for the purpose of tuning the hyperparameters. The integration of GRU, on the other hand, had to be conducted from scratch. For this, the *recommenders* repository [9] was used as a foundational resource.

NRMS. The strucuture of the model implemented as challange baseline includes two primary components: a news encoder and a user encoder, each employing multi-head self-attention layers to tracing complex patterns in the data. The news encoder processes article text by initially embedding it using pre-trained embeddings in the initial

implementation from *FacebookAI/xlm-roberta-base*, then refining these embeddings with self-attention to enhance contextual understanding and semantic connections within the text. Concurrently, the user encoder analyzes the user's click history, utilizing self-attention to trace the evolution of user interests over time. The encodings from the user and news are combined via a dot product to calculate relevance scores, which are then processed through softmax for multiple article predictions or sigmoid for individual predictions. Training involves a cross-entropy loss function and Adam optimizer to facilitate effective optimization. Evaluation as implemented includes the projects given metrics like AUC, MRR, and nDCG for accuracy.

GRU. In contrast to the other two models, the GRU was not implemented in the *ebnerd-benchmark* repository [5]. We initially attempted to fit the GRU implementation from the Microsoft recommenders library [9] to our needs. But this turned out to be surprisingly time-consuming as the library is both poorly documented and would have ultimately required changing most files. We were left on our own to figure out all the necessary fields for the iterator and the input for the GRU model. Additionally, it required more effort than anticipated to find out the logistics of reading, saving and passing files to functions as well as writing the final submission file.

On top of updating the outdated version of the provided GRU model in "gru.py", we preprocessed the data by exploding the dataframe along the inview id column as well as the article id column, before adding and generating further columns. We manually added lables to mark positive (=clicked) and negative (=viewed but not clicked) articles. Given that the GRU is not a time aware model, we padded two time specific columns with placeholder values, based on this advice provided by the authors of the library [8]: "Only the "SLi_Rec" model is time-aware. For the other models, you can just pad some meaningless timestamp in the data files to fill up the format, the models will ignore these columns".

Additionally, we have experimented with keeping the timestamps and negative sampling. Eventually we settled for having 4 negative samples in the train and validation file have which are not taken randomly out of all possible articles, but rather out of those which the user viewed but did not click on to improve the models results. When we added negative samples to the test file for evaluation purposes, we could only predict scores for both positive and negative instances. This resulted in five scores when adding four negative samples. However, since we aim to predict scores for all in-view articles of an impression ID to create the final submission file, we did not add negative samples to the test file to mirror the test set of the official challenge.

To summarize and put it in terms of the original data: the final prediction rankings are based on the pure validation dataset without any negative samples, while the model's evaluation is performed on the validation split of the training dataset, as this is necessary to have a ground truth for assessment.

LSTUR. Being initiated with hyperparameters and utilizing a pretrained word embedding matrix, the LSTUR implementation is incorporating GRU networks to adeptly handle sequential data. The user encoder melds long-term user embeddings, created from an embedding layer that converts user indexes into dense vectors, with short-term user representations produced by applying a GRU to the sequence of clicked article titles. The approach to user representation is adaptable, allowing for either initialization of the GRU with long-term embeddings or a concatenation of long and short-term embeddings, contingent upon the specified type ('ini' or 'con'). Simultaneously, the news encoder uses a convolutional layer to derive local features from article titles, complemented by an attention layer that accentuates key words, thereby enriching article representation.

The LSTUR model's structure incorporates inputs for user history, user indexes, and titles of articles under prediction, creating a thorough framework for modeling user-news interactions. The primary prediction method calculates the

dot product between user and news representations, employing softmax activation for multiple article predictions and sigmoid activation for single article predictions. Training this model involves strategies such as early stopping, model checkpointing, and performance metrics, all directed towards enhancing prediction accuracy while ensuring computational efficiency.

3.4 Parameter Optimization

For the purpose of model optimization, an approach was adopted that considers the models in a comprehensive manner. This involved varying not only the available document embeddings (not done for GRU as not applicable to the concept) but also model-dependent parameters such as model architecture, training, and data preprocessing. The embeddings tested through a grid search like approach were sourced from *Hugging Face* and included: *facebook/xlm-roberta-base*, *google-bert/bert-base-multilingual-cased* and *KennethTM/bert-base-uncased-danish*, to focus on the Danish language specifically. A selection of other tuned aspects includes (depending on the model) history size, head number and dimension, learning rate, activation function, dropout, title size and window size. See section 4 for a full optimization workflow.

As a technical solution for conducting hyperparameter optimization, *TensorFlow Keras Tuner* was selected, employing the *Hyperband* algorithm by Li [7]. This approach combines the advantages of lower resource utilization compared to Bayesian Optimization, greater time efficiency than pure grid search, and a more targeted exploration than random search. The choice was made with consideration of limited resource availability.

3.5 Testing and Evaluation

The datasets designated for completing the challenge (demo, small and large) were provided in various column-oriented parquet files. One subset is intended for training the models (train), and another for testing/validation (validation). Additionally, there is a dedicated test set intended for final evaluation. However, due to the described performance limitations, work was conducted solely with the small dataset. The training portion of the data was further divided for training and validation purposes, with the validation set used for testing. It is noteworthy that work was consistently performed on the entire data instance (fraction: 1.00), not the initially defined 1 %.

Model evaluation was performed using a comprehensive set of key accuracy metrics, such as Area Under the ROC Curve (AUC), Mean Reciprocal Rank (MRR), Normalized Discounted Cumulative Gain at K (used as nDCG@5 and nDCG@10), and validation loss. The AUC metric, implemented using scikit-learn, quantifies a model's ability to differentiate between classes at various thresholds by calculating the area under the receiver operating characteristic curve. The MRR metric is determined by arranging the predicted scores in descending order, locating the position of the first relevant item, and calculating the reciprocal of this rank, highlighting the significance of accurate early recommendations. The nDCG@K metric evaluates the quality of the top K recommendations by calculating gains for relevant items and applying a logarithmic discount based on their ranked positions, emphasizing the importance of higher ranking for relevant items. Validation loss, generally computed using cross-entropy during training, quantifies the discrepancy between predicted outcomes and actual labels, indicating the model's accuracy. Already in the base repository, these metrics were integrated into functions that analyze both the actual labels and predicted scores. Beyond Accuracy Metrics are deliberately excluded in this section.

4 Results

In this section, all empirical results are presented in numbers and visualizations. These are then analyzed and discussed in relation to the characteristics of the various models. We submitted our results through the contest's evaluation pipeline, yet they are not visible on the leaderboard. Our developments were consistently executed on the small dataset, as detailed in the report for performance reasons. According to the contest rules, focusing on a custom test set (in our case "small") is permissible. However, for complete participation on the leaderboard, predictions on the entire test set are required (see referring posts in forum).

For clarity, a complete tabular listing of the quality metric results including all values of tuned parameters has been omitted from the report. For GRU the only parameters tuned were: Hidden Size, Embedding Layer 1 and Embedding Layer 2 Please refer to the CSV file in the repository at "/gr26_eval_output/results" for these detailed outcomes. Table 1 present the challange's metrics of the three used models, for the initial configuration (fixed), optimized (hp_opt) and different document embeddings (in brackets). Note that usage of the pretrained model (loading saved weights) of the hand-in might lead to slightly different results due to precision limitations.

model_name	auc	mrr	ndcg@5	ndcg@10
GRU_fixed	0.4981	0.4819	0.6083	0.6083
GRU_hp_opt	0.5356	0.5076	0.628	0.628
LSTUR_fixed (FacebookAI-xlm-roberta-base)	0.5783	0.3614	0.4039	0.4789
LSTUR_fixed (google-bert-bert-base-multilingual-cased)	0.5706	0.3595	0.4007	0.4758
LSTUR_fixed (KennethTM-bert-base-uncased-danish)	0.5739	0.3609	0.4037	0.4773
LSTUR_hp_opt (FacebookAI-xlm-roberta-base)	0.5732	0.3630	0.4049	0.4791
LSTUR_hp_opt (google-bert-bert-base-multilingual-cased)	0.5698	0.3599	0.4012	0.4761
LSTUR_hp_opt (KennethTM-bert-base-uncased-danish)	0.5665	0.3538	0.3958	0.4709
NRMS_fixed (FacebookAI-xlm-roberta-base)	0.5553	0.3496	0.3896	0.4672
NRMS_fixed (google-bert-bert-base-multilingual-cased)	0.5505	0.3450	0.3844	0.4622
NRMS_fixed (KennethTM-bert-base-uncased-danish)	0.5495	0.3372	0.3775	0.4556
NRMS_hp_opt_test (FacebookAI-xlm-roberta-base)	0.5687	0.3585	0.3993	0.4743
NRMS_hp_opt_test (google-bert-bert-base-multilingual-cased)	0.5649	0.3582	0.3988	0.4742
NRMS_hp_opt_test (KennethTM-bert-base-uncased-danish)	0.5524	0.3448	0.3846	0.4621

Table 1. Performance Metrics of GRU, LSTUR and NRMS

Special attention was given to examining the metrics throughout the progression of epochs. Figure 1 displays the results for each metric and model configuration.

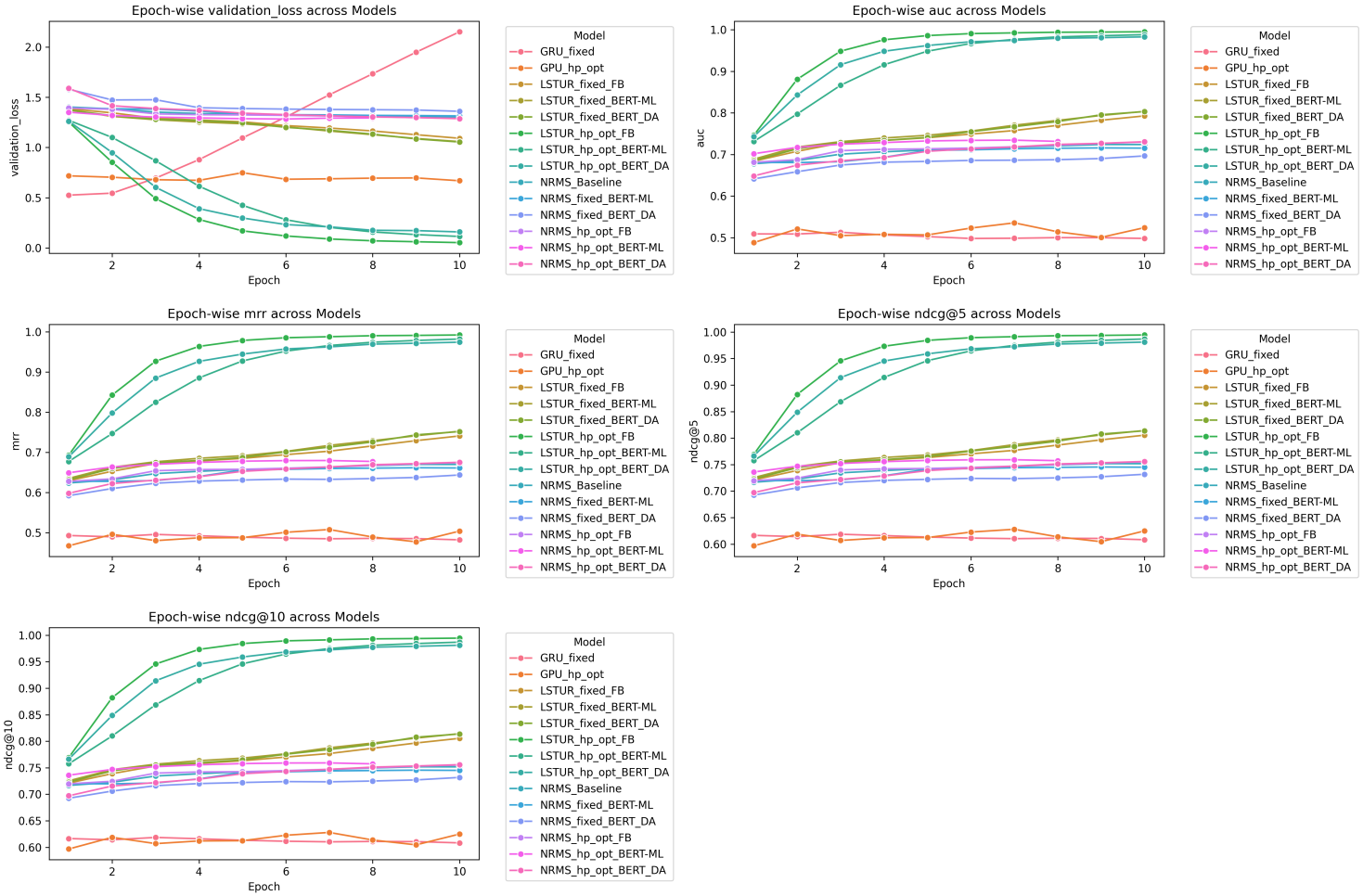

Fig. 1. Metrics of the Models over Epochs

4.1 Optimization Findings

During our hyperparameter optimization of the GRU, NRMS, and LSTUR models, we made several discoveries that enhanced our understanding of model performance and parameter sensitivity. The GRU model showed that its fixed configuration was nearly as effective as the optimized one, achieving an AUC of 0.4981, MRR of 0.4819, and NDCG@10 of 0.6083, compared to the optimized scores of AUC 0.5356, MRR 0.5076, and NDCG@10 0.628. This suggests the default parameters were already strong, though fine-tuning could marginally improve outcomes. Key factors such as attention hidden dimension and dropout rates were influential, impacting the model's ability to detect complex user patterns and preventing overfitting.

For the NRMS model, hyperparameter optimization had a more notable effect. Adjusting learning rates and dropout settings significantly enhanced the model's ability to generalize, improving metrics like AUC, MRR, and NDCG. With optimized settings using "FacebookAI-xlm-roberta-base" embeddings, NRMS recorded an AUC of 0.5687, MRR of 0.3585, and NDCG@10 of 0.4743, an improvement over the baseline scores of AUC 0.5553, MRR 0.3496, and NDCG@10 0.4672. The choice of document embeddings was crucial, with "FacebookAI-xlm-roberta-base" and "google-bert-base-multilingual-cased" consistently delivering superior performance. The "KennethTM-bert-base-uncased-danish" embeddings, tailored for Danish content, did not enhance performance as effectively, likely due to less comprehensive pre-training.

Similarly, the LSTUR model's performance benefited from the choice of document embeddings, particularly the "FacebookAI-xlm-roberta-base" which led to an AUC of 0.5732, MRR of 0.3629, and NDCG@10 of 0.4791. This indicates that embeddings from larger, more diverse datasets capture more detailed semantic information essential for effective news recommendation. The "google-bert-base-multilingual-cased" embedding also performed well with slightly lower metrics. Critical parameters for LSTUR included the model's history size and attention hidden dimension, with shorter history sizes providing more accurate short-term recommendations and longer dimensions enhancing the understanding of complex user behaviors.

Our optimization efforts highlighted the significance of initialization methods and learning rate schedules. Models initialized with a truncated normal distribution and using a learning rate decay strategy converged faster and yielded better performance. Additionally, balanced dropout rates prevented overfitting and ensured consistent performance across various validation sets.

Figure 1 illustrates the epoch-wise development of various metrics across different models and their configurations, providing valuable insights into the training dynamics and performance improvements over time. The validation loss graph indicates a general trend of decreasing loss with increasing epochs for most models, which suggests effective learning and convergence. However, an exception is noted in the GRU model, which exhibits fluctuating and increasing loss, indicating potential overfitting or instability.

In terms of AUC, the LSTUR and NRMS models, particularly when configured with optimized settings and advanced embeddings like "FacebookAI-xlm-roberta-base" and "google-bert-base-multilingual-cased," demonstrate a steep and consistent rise. This improvement reflects their enhanced ability to discriminate between relevant and non-relevant recommendations as training progresses. The MRR and NDCG@5 metrics further support these findings, showing that the LSTUR and NRMS models achieve higher reciprocal ranks and better ranking positions for top recommendations, respectively, compared to their fixed configurations and the GRU model. Additionally, the NDCG@10 metric underscores the effectiveness of the optimized models in maintaining high-quality recommendations, even when considering a broader set of items.

4.2 Discussion

The GRU model, designed to capture temporal dependencies in user behavior, showed stable performance across both fixed and optimized configurations. This stability suggests an effective initial setup with crucial parameters such as attention hidden dimensions and dropout rates being significant. The minor enhancements achieved through tuning underscore the GRU's potential for refinement, although the default settings already appear nearly optimal for this dataset.

In contrast, the LSTUR model demonstrated greater sensitivity to changes in hyperparameters and the selection of document embeddings. This model, aimed at balancing long-term and short-term user preferences, inherently requires more nuanced optimization. The marked performance boost with "FacebookAI-xlm-roberta-base" embeddings highlights the critical impact of high-quality, pre-trained embeddings on model effectiveness. LSTUR's ability to leverage detailed contextual information from these embeddings is evident in its improved metrics, illustrating its comprehensive use of diverse data features.

The distinctions between the LSTUR and GRU models reflect their architectural differences. LSTUR's design, which integrates both long-term and short-term user data, provides a broader context for understanding user behavior—essential for news recommendation systems where interests are varied and evolve. This architectural depth explains why LSTUR benefits more from advanced embeddings and fine-tuning compared to the GRU, whose simpler structure may not capture complex semantic contexts as effectively, resulting in less pronounced improvements.

Additionally, the NRMS model's performance gains underscore the importance of attention mechanisms in recommendation systems. NRMS, which primarily utilizes self-attention to discern relationships between news items, achieved significant enhancements with optimized settings that favor generalization and minimize overfitting.

These findings highlight the vital role of document embeddings, especially those trained on large, diverse datasets. The exceptional performance of the "FacebookAI-xlm-roberta-base" and "google-bert-base-multilingual-cased" embeddings across NRMS and LSTUR models attests to the capacity of rich, contextual embeddings to elevate a model's predictive prowess. Conversely, the "KennethTM-bert-base-uncased-danish" embedding, despite its language-specific design, underperformed, suggesting that comprehensive pre-training may outweigh the benefits of linguistic customization.

5 Conclusion

In this project, we addressed the 2024 RecSys Challenge by leveraging the Ekstra Bladet News Recommendation Dataset (EB-NeRD) to predict user engagement with news articles. Our strategy entailed deploying three distinct models: NRMS, GRU, and LSTUR, each designed to utilize various facets of user interaction data. Throughout the project, we encountered and surmounted considerable technical hurdles, such as high memory requirements and the need for efficient data preprocessing. Our efforts underscored the value of comprehensive documentation, but showed the benefit of being able to build uo on existing frameworks to meet specific requirements.

Despite the limitations posed by constrained resources, our models were able to compute performance enhancements through hyperparameter tuning and the strategic employment of document embeddings. The NRMS model excelled in capturing semantic subtleties through advanced self-attention mechanisms, while the LSTUR model skillfully managed the interplay between long-term and short-term user preferences. Although implementing the GRU model presented challenges, it yielded profound insights into patterns of temporal user behavior.

We assessed our results using established metrics like AUC, MRR, and nDCG, which revealed distinct advantages of each model in various facets of recommendation quality. Overall, this initiative not only enhanced our comprehension of

sophisticated recommendation algorithms but also highlighted the essential roles of adaptability and ongoing learning in addressing intricate data science challenges. The insights gained equip us to better tailor recommender system applications in real-world settings, enhancing user engagement and satisfaction.

Analyzing the results from our models provided key insights into building effective recommendation systems. The GRU model's performance stability underlines the efficacy of its initial parameter settings, demonstrating its capability to accurately capture temporal user behaviors. In contrast, the LSTUR model displayed a marked responsiveness to hyperparameter adjustments, significantly benefiting from high-quality document embeddings. This sensitivity highlights LSTUR's ability to effectively balance long-term and short-term user preferences, showcasing its adaptability and depth in interpreting complex user interactions.

The NRMS model also illustrated notable improvements when optimized, particularly through its use of self-attention mechanisms. The enhancements seen with precise tuning of learning rates and dropout settings emphasize the importance of such adjustments for improving generalization capabilities and preventing overfitting. Across all models, document embeddings, especially those pre-trained on diverse and extensive datasets, were crucial. These embeddings substantially bolstered the models' abilities to discern semantic nuances, thereby elevating their predictive accuracy. These findings not only refine our understanding of model dynamics but also underscore the critical role of advanced, contextual data processing in enhancing recommendation systems.

6 Organizational Details

In this section the workload distribution between the group members is noted. Additionally, a general lessons learned section is posed to present internal learnings made over the project.

6.1 Distribution of Workload

Work on the project was conducted in close, direct coordination, utilizing appropriate tools beyond mere tracking through git commits (commit numbers might be misleading). A breakdown of the distribution of work hours according to thematic assignments is as follows:

- Pia Schwarzinger: GRU Model Implementation (own implementation based on *recommenders*)
- Maximilian Ranzinger: Technical Setup, Baseline Setup, Hyperparameter Optimization (incl. Code Adaption)
- Philipp von der Thannen: NRMS Implemenation, LSTUR Implementation, Discussion and Analysis
- Yahya Jabary: Technical Setup, LSTUR implementation, NRMS implementation [1]

All team members contributed to the literature review and report writing.

6.2 Project Learnings

In the process of integrating into an existing project or codebase of the *ebnerd-benchmark*, the importance of high-quality documentation became apparent. The limited in-code documentation significantly delayed the development of an understanding of the foundational project. However, it also highlighted how valuable existing building blocks can be for progress.

The limited technical resources posed a challenge throughout the project. On one hand, they inherently restricted the execution of recommendation systems to datasets of a certain size; on the other hand, they were simply not efficient

[1] On Github, to enable Condabench code submission: https://github.com/sueszli/repetitive-neural-nonsense

enough to consistently ensure a reasonable runtime. It is necessary to find a balance between performance and stability. Therefore, it is important to generate intermediate results and consistently save them persistently.

It was intriguing to see how specific models are implemented in Python and how their performance varies. It is likely that the distinct characteristics of the prediction models would become even more apparent on larger datasets.

References

[1] Mingxiao An, Fangzhao Wu, Chuhan Wu, Kun Zhang, Zheng Liu, and Xing Xie. 2019. Neural News Recommendation with Long- and Short-term User Representations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Anna Korhonen, David Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, Florence, Italy, 336–345. https://doi.org/10.18653/v1/P19-1033

[2] Ekstra Bladet. 2024. Ekstra Bladet News Recommendation Dataset. https://recsys.eb.dk/dataset/. [Online; accessed 24-June-2024].

[3] Ekstra Bladet. 2024. RecSys Challenge 2024. https://recsys.eb.dk. [Online; accessed 28-June-2024].

[4] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential User-based Recurrent Neural Network Recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* (Como, Italy) *(RecSys '17)*. Association for Computing Machinery, New York, NY, USA, 152–160. https://doi.org/10.1145/3109859.3109877

[5] Kruse Johannes. 2024. ebnerd-benchmark: Ekstra Bladet Recommender System repository for benchmarking the EBNeRD dataset. https://github.com/ebanalyse/ebnerd-benchmark

[6] Marius Kaminskas and Derek Bridge. 2016. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Trans. Interact. Intell. Syst.* 7, 1, Article 2 (dec 2016), 42 pages. https://doi.org/10.1145/2926720

[7] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. http://arxiv.org/abs/1603.06560

[8] Microsoft. 2024. Microsoft Recommenders Algorihms – GRU tutorial. https://github.com/recommenders-team/recommenders/blob/main/examples/00_quick_start/sequential_recsys_amazondataset.ipynb. [Online; accessed 24-June-2024].

[9] recommenders team. 2024. recommenders: Best Practices on Recommendation Systems. https://github.com/recommenders-team/recommenders

[10] Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang, and Xing Xie. 2019. Neural News Recommendation with Multi-Head Self-Attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 6389–6394. https://doi.org/10.18653/v1/D19-1671