RecSys Challenge 2024
WeKnowWhatYouWant

STEFAN HAMM, TU Wien, Austria
PETAR SORIC, TU Wien, Austria
EVA PLEŠE, TU Wien, Austria
WOJCIECH MICHALUK, TU Wien, Austria
MARKO GUGLETA, TU Wien, Austria

WeKnowWhatYouWant presents our participation in the ACM RecSys 2024 Challenge using the Ekstra Bladet News Recommendation Dataset (EB-NeRD). This dataset, gathered from Danish news site Ekstra Bladet, contains user interaction logs over six weeks and includes comprehensive article metadata. Our project employs two state-of-the-art recommendation algorithms: Neural Recommendation with Personalized Attention (NPA) and Neural Recommendation with Long- and Short-term User Representations (LSTUR).We aim to enhance these recommender models to predict user preferences effectively. The project's evaluation follows stringent metrics like AUC, MRR, and nDCG@K, ensuring our solutions meet high standards. Our results will be benchmarked and submitted to the RecSys Challenge leaderboard, with detailed documentation and reproducibility ensured through comprehensive coding and reporting practices.

1 Introduction

The ACM RecSys 2024 Challenge gives us a great opportunity to apply our knowledge on recommender systems using the Ekstra Bladet News Recommendation Dataset (EB-NeRD). This dataset includes user interaction logs from the Danish news site Ekstra Bladet, collected over six weeks from April 27 to June 8, 2023. It contains detailed information about the news articles such as titles, abstracts, bodies, categories, and more. Our task is to develop algorithms that can accurately predict which articles users will engage with, which is a complex and interesting problem.[3]

Our group, WeKnowWhatYouWant, used two algorithms for this challenge: the Neural news recommendation with Personalized Attention (NPA) and the Long-Short Term User Representation (LSTUR). NPA is a news recommendation model that uses a CNN for news representation based on article titles, learns user representation from clicked articles, and employs personalized attention at both word and news levels to capture user-specific informativeness, while LSTUR is a news recommendation approach that captures both long-term and short-term user preferences using a news encoder to learn from titles, a user encoder for long-term preferences via user ID embeddings, and a GRU network for short-term preferences from recently browsed news, with two methods to combine these representations [2].

First, we implemented the baseline model provided in the EB-NeRD repository. Then, we worked on integrating our chosen algorithms. Our approach includes data preprocessing, feature engineering, and model optimization to boost

Authors' Contact Information: Stefan Hamm, TU Wien, Vienna, Austria, e12346011@student.tuwien.ac.at; Petar Soric, TU Wien, Vienna, Austria, e12340235@student.tuwien.ac.at; Eva Pleše, TU Wien, Vienna, Austria, e12341572@student.tuwien.ac.at; Wojciech Michaluk, TU Wien, Vienna, Austria, e12341799@student.tuwien.ac.at; Marko Gugleta, TU Wien, Vienna, Austria, e12016483@student.tuwien.ac.at.

Manuscript submitted to ACM

the performance of our recommendations. We evaluated our models using common metrics like the area under the ROC curve (AUC), mean reciprocal rank (MRR), and normalized discounted cumulative gain (nDCG@K). Additionally, we considered beyond-accuracy objectives such as diversity, serendipity, novelty, and coverage to make sure our recommendations are not only accurate but also diverse and engaging.

This project is a great chance for us to apply what we've learned during the course of Recommender Systems.

2 Data Description

The Ekstra Bladet News Recommendation Dataset (EB-NeRD) was created to facilitate research in news recommendation systems, derived from user behavior logs collected at Ekstra Bladet, a Danish news platform. The dataset covers a 6-week period from April 27 to June 8, 2023, focusing on active users who engaged with news articles during this timeframe.

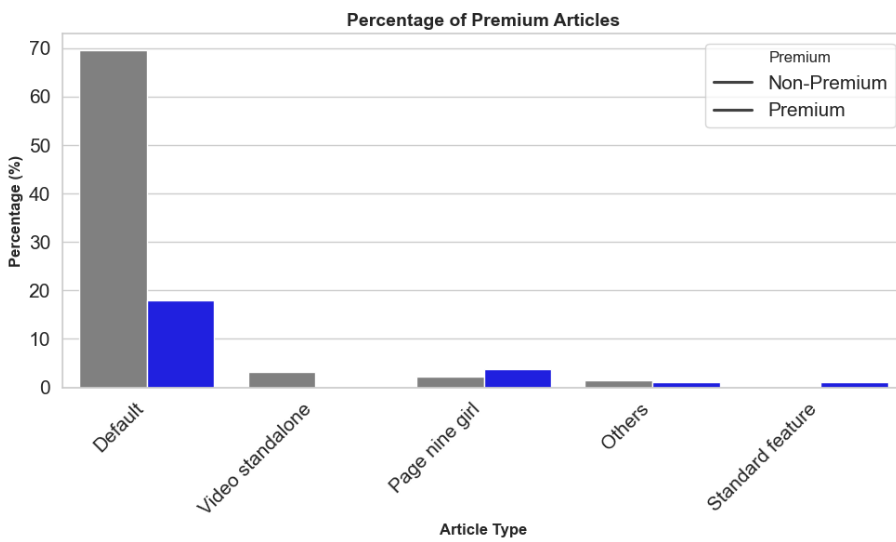

Fig. 1. Premium articles vs regular articles

After analysis of the given data, our team observed some dependencies that might be useful for predictions. The ratio of premium to non-premium articles is presented in Figure 1. It shows a significant dominance of regular articles by default. In other categories, there are more premium articles. Since articles that are premium can't be recommended to non-premium users, this limits the articles and increases the difficulty of recommendation slightly.

2.0.1 *Datasets.*
The dataset is divided into demo, small and large subsets, each of which is further split into training and validation sets. Each set contains two Parquet files: one for user behavior and one for user history, representing the behavior and history of a given user.

(1) **Impression Logs (`behaviors.parquet`):**
- **Impression ID**: Unique identifier for each impression.
- **User ID**: Anonymized user identifier.
- **Article ID**: Unique ID for the news article displayed in the impression; empty if from the front page.

- **Session ID**: Identifier for the user's browsing session.
- **Inview Article IDs**: List of news articles seen by the user in the impression.
- **Clicked Article IDs**: List of news articles clicked by the user in the impression.
- **Time**: Timestamp of the impression.
- **Readtime**: Time spent by the user on the page.
- **Scroll Percentage**: Percentage of the article scrolled by the user.
- **Device Type**: Type of device used (desktop, mobile, tablet, or unknown).
- **SSO Status**: Indicates if the user logged in via Single Sign-On.
- **Subscription Status**: User's subscription status (paid subscriber or not).
- **Gender**, **Postcode**, **Age**: Demographic information of the user.
- **Next Readtime**, **Next Scroll Percentage**: Metrics for the next article the user clicked.

(2) **User Click Histories (`history.parquet`):**
- **User ID**: Anonymized user identifier.
- **Article IDs**: List of articles clicked by the user.
- **Timestamps**: Timestamps of when the articles were clicked.
- **Read times**: Time spent reading each article.
- **Scroll Percentages**: Scroll percentages for each article.

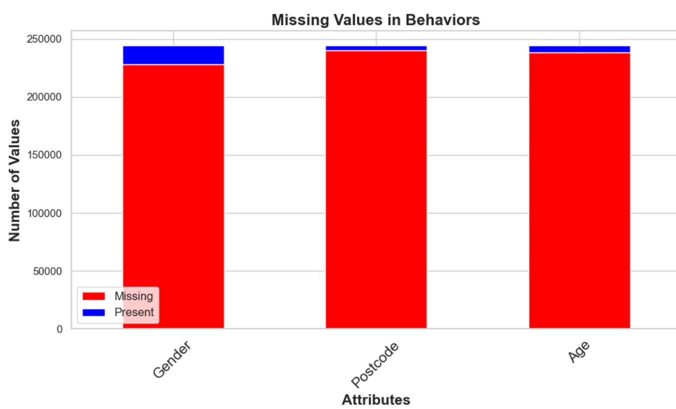

Fig. 2. Missing data behaviours

Figure 2 illustrates the number of missing values in the Behaviours dataset. In some columns which could be used for predictions, such as "Age," it is above 90%. This makes those columns unsuitable for predictions. Undoubtedly, any data imputation technique can be difficult to implement with such a high level of missing data.

2.0.2 Articles.

- **News Articles (`articles.parquet`):**
  - **Article ID**: Unique identifier for the article.
  - **Title**, **Subtitle**, **Body**: Danish text providing the article's content.
  - **Category ID**, **Category String**: Categorization of the article.

– **Subcategory IDs**: Subcategories associated with the article.
– **Premium**: Indicates if the content is behind a paywall.
– **Time Published**, **Time Modified**: Timestamps for article publication and modification.
– **Image IDs**: IDs of images associated with the article.
– **Article Type**: Type of article (e.g., feature, gallery, video).
– **URL**: URL of the article.
– **NER (Named Entity Recognition)**, **Entities**: Tags from proprietary models based on article text.
– **Topics**: Tags from a proprietary topic-recognition model.
– **Total Inviews**, **Total Pageviews**, **Total Read Time**: Metrics on user interactions with the article within the first 7 days after publication.
– **Sentiment Score**, **Sentiment Label**: Sentiment analysis results based on article title and abstract.

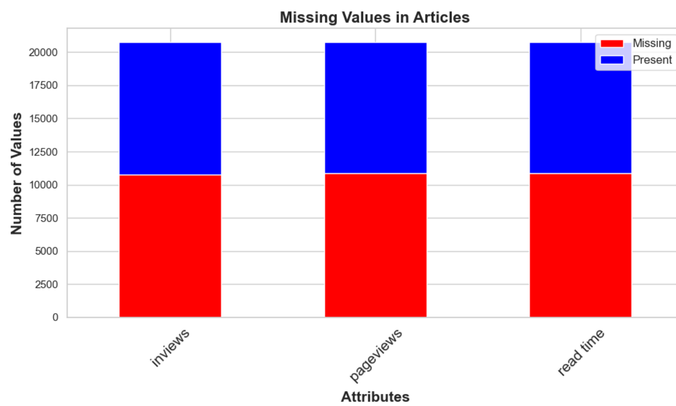

Fig. 3. Missing data articles

Figure 3 shows that in the Articles dataset, the percentage of missing data in columns is maximally around 50%. This makes those features hard to use in predictions. Again, with such a level of missing values, data imputation techniques could be implemented.

2.0.3 Artifacts. Additional data includes:

- **Article Embeddings**: Continuous representations of articles derived from textual content.
- **Image Embeddings**: Encoded representations of article images.

The dataset is divided into demo, small, and large bundles, each containing training and validation sets, with a separate test set available.

3 Approach

The given ebnerd-benchmark repository [1] was checked out to understand the baseline model (NRMS) provided by the organizers. Two models were selected to implement:

- Long- and Short-term User Representations (LSTUR)(chosen)
- Personalized Attention (NPA)(assigned)

Using the provided examples, the data loaders were set up. The data was downloaded in zip format using the file `downloadfiles.py` and unpacked. The data was then preprocessed with `data_pre_process.ipynb`, which prepares the input data for the algorithm. This script reads and merges the behavior and history Parquet files, preparing the dataset for training and validation. Additionally, it retains only a subset of columns (DEFAULT_USER_COL, DEFAULT_HISTORY_ARTICLE_ID_COL, DEFAULT_INVIEW_ARTICLES_COL, DEFAULT_CLICKED_ARTICLES_COL, DEFAULT_IMPRESSION_ID_COL) to improve algorithm efficiency. Finally, it preprocesses the test set and samples the first 10K rows for later predictions. All algorithms used in the following analyses were applied to this preprocessed dataset. Training and validation were performed on the small dataset. For the reasoning behind this approach and using only 10K rows for prediction, see section 3.2.

For both `lstur` and `nrms`, a HuggingFace Transformer model and tokenizer (`xlm-roberta-base`) were initialized[5].
In `lstur`, text columns in the dataset were processed using the transformer to convert them into token encodings and mappings were created from article IDs to their token representations. User ID mappings were also created for unique users in the training dataset. The LSTUR model was configured and initialized, trained using the training DataLoader, and evaluated on the validation DataLoader.

Regarding the NPA algorithm implementation, the same data loading setup was used as mentioned above, including the transformer model and tokenizer. Using the already implemented model and dataloader from the ebnerd repo, NPA was first trained on small dataset then on large dataset for 1 epoch only, because of resource and time constraints. Validation set was used to calculate metrics, test set was used for generating predictions.

In `nrms`, the NRMS model was initialized and trained using the training DataLoader and evaluated on the validation DataLoader. The best model weights were loaded based on validation performance. Predictions were made on the test dataset, and performance was evaluated using metrics such as AUC, MRR, and NDCG.

At the end of the experiments, predictions were generated on parts of the test dataset and the predictions were written to a submission file for further use or evaluation in the format required by RecSys Challenge 2024.

3.1 Improvements of the algorithms

Running the algorithms on the full dataset proved challenging, so our primary focus was on achieving functionality with the models. After testing all three models on a smaller dataset and obtaining initial results, NPA demonstrated better performance over both the baseline and lstur, particularly in MRR, NDCG@5, and NDCG@10. Consequently, we concentrated on optimizing NPA further. We attempted to improve its performance through hyperparameter tuning using Keras Tuner, but encountered frequent crashes that prevented successful completion of the process.

3.2 Problems

While implementing the given and chosen algorithms, several problems were encountered. Some of them were resolved, while others remained as limitations.

RAM The Baseline nrms implementation in the ebnerd-benchmark repository [1] is quite inefficient. Running the baseline Jupyter notebook often crashes with large datasets or shows an estimation time of multiple days on test sets. The main issue is the inefficient dataloader, which uses more than 120 GB of RAM. This exceeds the capacity of most student laptops, typically equipped with at most 32 GB RAM, causing frequent crashes. Some solutions include reducing batch size or, as done by other groups, rewriting the entire code. This is one of the main reasons why we switched to the small dataset.

ebnerd-benchmark Repo Committing to the repository's dataloaders hindered significant progress. As students, we expected an efficient baseline to build upon. Instead, our algorithms required hours to train and predict, even on the cluster. We realized too late that the inefficient ebnerd-benchmark implementation was the root cause, and rewriting the code from scratch in PyTorch was unfeasible due to time constraints.

Resources The cluster also struggled with the workload. Running models/predictions with over 120 GB RAM usage would cause execution to stop without errors. Having multiple groups running this inefficient code just amplified the problem. The CPU-only servers could not run the preprocessing step where the full test set is loaded into a Polars dataframe, causing the server to crash. The preprocessing only worked on GPU servers.

Test Set Merging the test set history with the test set behaviors parquet file results in about 13.5 million entries. Predictions on the full test set using the given dataloaders took multiple days and suffered from random crashes. The last few rows, containing more information, further amplified this issue, making full test set predictions impossible. We resorted to sampling the test set to about 10k rows with a reduced batch size to obtain at least any predictions at all. Another solution would be to split the test set into chunks of 500k rows and run predictions on each chunk. This would help avoid losing any predictions while processing the test set. However, this approach might slightly increase prediction time due to additional loading and writing times.

Repeatability on Cluster and local To ensure repeatability, a conda environment as described in ebnerd-benchmark was created [1]. This environment was used both locally and on the cluster. However, the previously trained weights worked locally but not on the cluster, despite using the same environment and code. This is possibly due to different cuda drivers installed on the machine where jupyter server is hosted, but debugging this would require taking a look and possibly modifying the host machines, which was beyond the scope of this project.

Beyond Accuracy Metrics Due to the mentioned issues with RAM usage, inefficient data loaders, and resource constraints, we were unable to generate predictions for the entire test set. This made it impossible to compute beyond accuracy metrics such as diversity, serendipity, novelty, and coverage. These metrics require full test set predictions to be meaningful, and our limited test set samples did not provide a reliable basis for these evaluations. The next step after acquiring test set predictions would be to load the test set with all the necessary extra columns for beyond accuracy metrics, create a candidate list and then use our trained model to add it's predictions for the candidate list, finally comparing it to baselines.

4 Results

The best results our team obtained were from the NPA model. It was trained for 10 epochs on a small dataset. When attempting to run on the entire available data, personal computers crashed. Nonetheless, it presents satisfying results compared to other groups. On validation dataset the accuracy was about 50%.

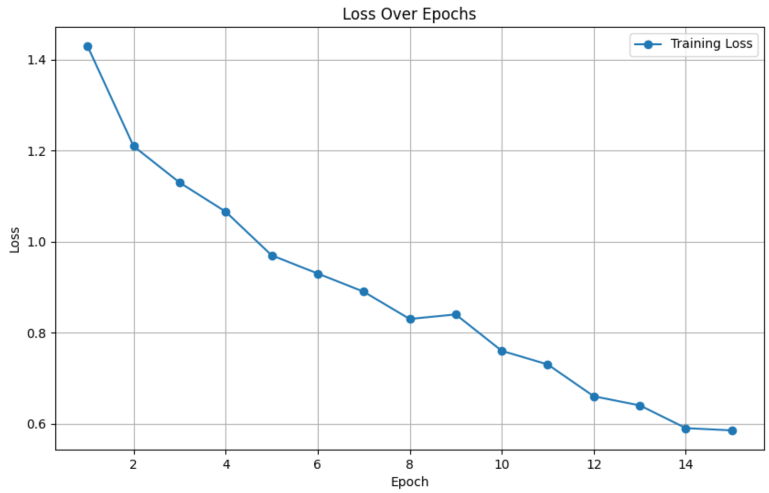

Fig. 4. NPA loss function over epochs

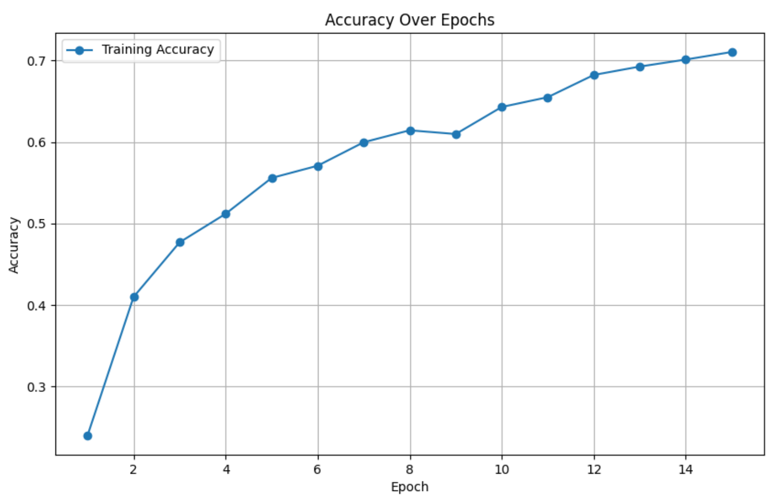

Fig. 5. NPA accuracy over epochs

The graph above presents a constant, smooth decrease in the loss function. We applied the early stopping technique to halt the training process when overfitting started. This is visible in the fig 4, as the training loss gradually decreases but the accuracy starts to decline on fig 5.

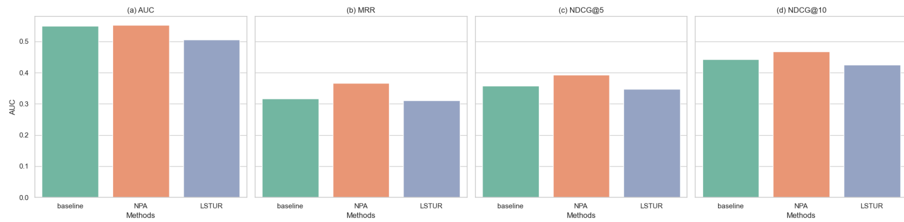

Fig. 6. Summary of metrics for validation

On the above figure 6, the baseline model, NPA, and LSTUR were compared in terms of:

- **AUC**: Area Under the Receiver Operating Characteristic Curve
- **MRR**: Mean Reciprocal Rank
- **NDCG**: Normalized Discounted Cumulative Gain

The NPA model shows a slight improvement over the baseline and LSTUR methods. For both the top 5 and top 10 items, LSTUR and the baseline have slightly worse predictions compared to the NPA model.

4.1 Beyond Accuracy Metrics

Beyond-accuracy metrics provide a more holistic evaluation of recommender systems, capturing aspects of user satisfaction that go beyond mere accuracy.

Diversity measures the variety of recommendations, ensuring that users are exposed to a wide range of items rather than similar ones. A diverse set of recommendations can increase user engagement by catering to different tastes and preferences. **Serendipity** refers to the ability of the recommender system to suggest unexpected items that are both relevant and pleasantly surprising to the user. This metric helps in keeping the user experience fresh and exciting. **Novelty** gauges the extent to which recommended items are new or previously unseen by the user, promoting exploration of new content. **Coverage** assesses the proportion of items in the catalog that are recommended across all users, indicating how well the system utilizes its entire inventory and ensures less popular items also get recommended.[6]

Unfortunately, we were not able to calculate these beyond-accuracy metrics because we couldn't obtain predictions for the entire test set due to the challenges in handling large datasets and inefficient data loaders.

4.2 Advantages and Disadvantages of Selected Methods

The following section examines the strengths and weaknesses of each algorithm used in this Group Project.

4.2.1 Neural Propagation Algorithm (NPA) [7]. **Advantages:**

- **Personalized Attention Mechanism:** NPA uses a personalized attention mechanism that captures the importance of different parts of the news content based on user preferences, which enhances the relevance of recommendations.
- **Effective for Textual Data:** It is particularly effective for datasets with rich textual information, as it can focus on specific content features that are significant to individual users.

- **Improved User Experience:** By focusing on personalized content, NPA can potentially improve the relevance of recommendations, leading to a better user experience.

Disadvantages:

- **Complexity:** The model's architecture is more complex compared to simpler recommendation algorithms, which can make it harder to implement and tune.
- **Resource Intensive:** NPA requires significant computational resources for training and inference, especially when dealing with large datasets.
- **Cold Start Problem:** Like many recommendation systems, NPA may struggle with the cold start problem where there is limited data on new users or items.

4.2.2 *Long- and Short-term User Representations (LSTUR)[4].* **Advantages:**

- **Captures Long- and Short-term Preferences:** LSTUR models both long-term user preferences (user's historical behavior) and short-term preferences (recent interactions). This dual approach can provide more accurate recommendations by considering both types of user behavior.
- **Adaptability:** It can quickly adapt to changes in user behavior by considering recent interactions, making it effective for dynamic environments where user preferences can change frequently.
- **Enhanced Personalization:** By integrating both long- and short-term preferences, LSTUR can deliver highly personalized recommendations.

Disadvantages:

- **Model Complexity:** LSTUR's architecture is complex due to the integration of both long- and short-term preferences, requiring careful tuning and substantial computational power.
- **Data Requirements:** It requires extensive historical and recent user interaction data to be effective. In scenarios with sparse data, its performance may degrade.
- **Cold Start Problem:** Similar to NPA, LSTUR can also face challenges with new users and items where there is insufficient interaction data to model preferences accurately.

Overall the Properties of NPA and LSTUR can summarized as seen in Table 1 [2].

Method	Properties
NPA	• NPA is a content-based news recommendation method. • It uses a CNN network to learn news representation. And it learns user representations from their clicked news articles. • A word-level personalized attention is used to help NPA attend to important words for different users. • A news-level personalized attention is used to help NPA attend to important historical clicked news for different users.
LSTUR	• LSTUR captures users' both long-term and short term preference. • It uses embeddings of users' IDs to learn long-term user representations. • It uses users' recently browsed news via GRU network to learn short-term user representations.

Table 1. Properties of NPA and LSTUR

5 Conclusion

During the experiments, the following conclusions have been noticed. Extensive lack of data in some key features, such as age, could be estimated based on data augmentation techniques. This might be a tough task due to the major lack of data, but it could be beneficial for predictions. Definitely, training on the full dataset would allow us to capture more patterns in the training data. This might be possible with better resources, such as a cluster of computers available for training, or rewriting the data loaders to be more performant.

The NPA model slightly outperforms the baseline and LSTUR. This is possible due to the personalized attention mechanism, which assumes that different users can have different attention distributions for the same news articles. This technique seems more suitable for the given task.

We encountered numerous challenges, such as limited RAM resources, extended training and prediction times, inefficient baseline model code, and managing an exceptionally large dataset relative to available computing resources. However, our team successfully applied the knowledge gained in the recommender systems course. This project enabled us to reflect on common issues encountered in the integration of machine learning models. It turned out that understanding system limitations is a crucial aspect when implementing recommender systems on big datasets. Moreover, the inability to generate predictions for the entire test set or run on the large dataset restricts us from generating more meaningful metrics, such as those beyond accuracy.

6 Workload distribution

Work Distribution

- **STEFAN HAMM:**
  - Intial setup + conda environment
  - Data preprocessing
  - LSTUR
  - run.sh
  - Final Report
- **PETAR SORIC:**
  - NPA implementation
  - NPA training and evaluation
  - Final presentation in class
- **EVA PLEŠE:**
  - Analysis of Dataset, graphics
  - Slides for final presentation
  - Final report
- **WOJCIECH MICHALUK:**
  - Baseline
  - Analysys of Dataset, graphics
  - Final Report
- **MARKO GUGLETA:**
  - NPA implementation
  - Hyperparameter tuning for NPA
  - Final presentation in class

References

[1] [n. d.]. ebanalyse/ebnerd-benchmark: Ekstra Bladet Recommender System repository for benchmarking the EBNeRD dataset. https://github.com/ebanalyse/ebnerd-benchmark. (Accessed on 07/04/2024).

[2] [n. d.]. recommenders-team/recommenders: Best Practices on Recommendation Systems. https://github.com/recommenders-team/recommenders?tab=readme-ov-file. (Accessed on 07/02/2024).

[3] [n. d.]. RecSys Challenge 2024. https://www.recsyschallenge.com/2024/. (Accessed on 07/01/2024).

[4] 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

[5] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised Cross-lingual Representation Learning at Scale. *CoRR* abs/1911.02116 (2019). arXiv:1911.02116 http://arxiv.org/abs/1911.02116

[6] Thomas E. Kolb, Irina Nalis, and Julia Neidhardt. 2023. Like a Skilled DJ - an Expert Study on News Recommendations Beyond Accuracy. In *Proceedings of the International Workshop on News Recommendation and Analytics co-located with the 2023 ACM Conference on Recommender Systems (RecSys 2023), Singapore, 18 September 2023 (CEUR Workshop Proceedings, Vol. 3561)*, Benjamin Kille (Ed.). CEUR-WS.org. https://ceur-ws.org/Vol-3561/paper6.pdf

[7] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. 2019. NPA: Neural News Recommendation with Personalized Attention. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining (KDD '19)*. ACM. https://doi.org/10.1145/3292500.3330665

A Appendix: Abstract Generation using ChatGPT

In this section, we provide an example of how we used ChatGPT to assist in generating the abstract for this paper. The following is a simulated Q&A interaction that showcases how ChatGPT was used.

A.1 ChatGPT Interaction

Question: Write the abstract for following task. Our group is name WeKnowWhatYouWant and our algorithms are npa and lstur.

Zum Hauptinhalt TUWEL EN DASHBOARD MEINE KURSE 2 TUWien logo 194.035-2024S Group Project Project Description

Project Description Abschlussbedingungen GROUP PROJECT (v08.04.2024) The group project is based on the ACM RecSys 2024 Challenge and the newly introduced Ekstra Bladet News Recommendation Dataset (EB-NeRD), which was collected from user behavior logs at Ekstra Bladet. This dataset covers the active users' interactions over a 6-week period from April 27 to June 8, 2023, and includes Danish news articles enriched with features such as title, abstract, body, categories, and more. The challenge for this year is to utilize this dataset to develop news recommendation algorithms that can effectively predict user engagement with articles. For more details, refer to the dataset format and specifics provided.

This year as part of this lecture we are participating as groups within the ACM RecSys 2024 Challenge. Please keep in mind, that this description is the single point of truth for the group project.

How to start? Carefully read through the group description Find group members and register for a group on TUWEL (define a group name + responsible person for communication with the lecture team, and send this information to us latest at 21st of April) Download the EB-NeRD dataset from the challenge website: https://recsys.eb.dk Register on the evaluation system: https://www.codabench.org/competitions/2469/?secret_key=98314b2c-9237-471e-905c-2a88bf6a1d8a (create an organization with your group name and invite your group members, the group name must be equal to the group name defined in step (2)) Accept the TUGitLAB invitation to your group-repository Have a look at the repository (https://github.com/ebanalyse/ebnerd-benchmark) provided by Ekstra Bladet Have a look at the different recommender system implementations (https://github.com/recommenders-team/recommenders) Submit a ranked preference list regarding the algorithms available within the recommenders repository (groups per algorithm is limited) on TUWEL (submission is starting on the 19th of April at 12:00). Setup your personal development environment you have the following options: provided JupyterHub environment (you have access to a GPU based notebook, these are currently within a beta state provided by the datalab) there is a shared group folder which you can use to exchange files with your team members working with google colab working locally (e.g. with Visual Studio Code and Anaconda) Make yourself familiar with the data by using the provided notebook (https://github.com/ebanalyse/ebnerd-benchmark/blob/main/examples/00_quick_start/nrms_ebnerd.ipynb) Implement the baseline and afterwards your assigned algorithm (see step (7); based on the recommenders repository) Implement an algorithm recent and fitting algorithm of your choice (can also be from the recommenders-github) Improve all your algorithms by incorporating aspects described within the challenge task Keep in mind the evaluation criteria listed here: https://recsys.eb.dk/ ("To address the normative complexities inherent in news recommendations, the test set incorporates samples specifically designed to assess models based on normative properties. This includes evaluating models on Beyond-Accuracy Objectives, such as intra-list diversity, serendipity, novelty, coverage, among others. The final result is the average of these metrics across all impression logs.") Look at the referenced paper for the beyond accuracy measures After evaluating and making initial improvements to all three methods, it is permissible to focus optimization efforts on one of the algorithms. This decision must be reasonably justified. NOTE: If it is not computationally feasible with the available computational resources you can perform training and validation on the Ebnerd-small dataset. However, if you use the small dataset explain what you tried in order to run the algorithms on the regular dataset and why it did not work. Test your implementation and upload your results of at least one approach, that performs the best in your opinion, to the benchmark website Package your final project (README.md, requirements.txt, run.sh, results.csv, source code, models, etc.) Submit your solution, on the provided group repository within TUgitLAB (do not forget the "final" Tag!) If required place all files which are too big into your group folder on JupyterHub (list them in your README.md) Upload your project report What is the task? Based on the task of the ACM RecSys 2024 Challenge you need to implement (= make use of the provided code by the challenge organizers as also the recommenders repository):

The baseline, given in the repository "NRMS on EB-NeRD" One recent and fitting algorithm out of the Recommenders repository (via TUWEL registration, groups per algorithm are limited) One recent and fitting algorithm of your choice Improve the selected approach to enhance your recommendation performance within the leaderboard (feature engineering, embeddings, data preprocessing, modifications of the algorithm, etc.) Group Formation Group formation will take place through TUWEL and should be completed by 14.04.2024.

Group sizes: 5 members Define a group name and contact person for communicating with the lecture team (send this information to us until the 21st of April*) All groups < 5 members will be merged into bigger groups on 15.4.2024!

Any questions? Ask on TUWEL or send us an mail!

*recsys-ss24@ec.tuwien.ac.at (subject: “[group] - <number_on_tuwel>”)

Evaluation “To evaluate the models we use several standard metrics in the recommendation field, including the area under the ROC curve (AUC), mean reciprocal rank (MRR), and normalized discounted cumulative gain (nDCG@K) for K shown recommendations. To address the normative complexities inherent in news recommendations, the test set incorporates samples specifically designed to assess models based on normative properties. This includes evaluating models on Beyond-Accuracy Objectives, such as intra-list diversity, serendipity, novelty, coverage, among others. The final result is the average of these metrics across all impression logs.” (https://www.recsyschallenge.com/2024/)

Document in your report how you improved the performance of your model by taking the evaluation criteria into account.

How to submit your solution? The submission consists of three parts:

Leaderboard: Where to submit?

You need to submit your solution to the RecSys Challenge leaderboard

Code: Where to submit?

You need to submit your project code to your assigned TUgitLAB repository by tagging the commit of your final submission with the word "final"

We collect solutions by this tag, if this tag is missing your solution cannot be graded!

Your submission must contain:

your source code (from all team members) Including a proper setup for reproducing your solution (requirements.txt, used python version, etc.) your models (if they are too big save them on JupyterHub within your group folder inside a folder called “submission” and reference them in the "README.md") a "README.md" which describes in detail how we can run your code (it is important that your solution is reproducible) a shell script "run.sh" which contains the relevant code to run the chosen algorithms which produces the final "results.csv" file, creates a conda environment including installing all dependencies, … (it is important that your solution is reproducible) If you want to submit multiple "results.csv" please name them "results_'methodname'.csv" ('methodname' is your self-defined name) Your solution must be self-contained, you can assume that we run your solution in an environment similar to the student JupyterHub service. All group members need to work on the implementation and commit their contributions to the repository on their own! (everyone needs to contribute to the repository)

Keep in mind that if we are not able to reproduce your predictions by using the provided code in the "run.sh" and the "README.md" explanation points will deducted.

If we are not at all able to reproduce and run your project there will be major point deductions which can result in failing the group project.

Submission Deadline: 28.06.2024

Project Report: Title page with a title of choice + group members Abstract (150 – 200 words) Sections Introduction (including your leaderboard group name) Data (description of the data) Approach (description of methods used, their implementation, performance measurements and comparison, improvements of your model on the leaderboard, justification of chosen algorithm, etc.) Results (empirical results, discussion of advantages and disadvantages of different methods) Conclusion References Organization of group work (who did what? / workload distribution) Max 10 pages (excl. title page, abstract, references and the organization)

Template: Prepare your report according to the latest single-column ACM template (acmart v2.05 or newer). Use the "manuscript" option with the \documentclass[manuscript]acmart to create a single-column format).

Additional relevant material as Appendix (your group submission results on the leaderboard, etc.)

Submission via TUWEL

Report as PDF ChatGPT for the Report?

Yes, but then you are required to put all used prompts and the corresponding answers into the appendix of the report

Please use the correct template as described above! There will be a severe deduction of points if the wrong template is used.

Submission Deadline: 28.06.2024

Helpful links: Link of the RecSys Conference: https://recsys.acm.org/recsys24/challenge/ https://www.recsyschallenge.com/2024/ Company Challenge Description: https://recsys.eb.dk/dataset/ Repository for Benchmarking: https://github.com/ebanalyse/ebnerd-benchmark Transformers Repository: https://github.com/recommenders-team/recommenders RecSys Leaderboard: https://www.codabench.org/competitions/2469/?secret_key=98314b2c-9237-471e-905c-2a88bf6a1d8a Challenge Forum (for questions related to the challenge): https://www.codabench.org/forums/2387/ Grading The grading will be done manually. Late submissions are not possible. We reserve the right to carry out automatic plagiarism checks. Both submission of your code and submission of the report are required! Last but not least, we want to see that you can apply the knowledge of this lecture. We know that technical knowledge varies, so we assess how well your project is carried out overall. Therefore please reserve time of your time budget for the group project to write "clean" code, to structure your code well and finally to write a well-structured project report!

Answer: Zuletzt geändert: Dienstag, 21. Mai 2024, 15:44 ChatGPT Abstract WeKnowWhatYouWant presents our participation in the ACM RecSys 2024 Challenge using the Ekstra Bladet News Recommendation Dataset (EB-NeRD). This dataset, gathered from Danish news site Ekstra Bladet, contains user interaction logs over six weeks and includes comprehensive article metadata. Our project employs two state-of-the-art recommendation algorithms: Neural Propagation Algorithm (NPA) and Long-Short Term User Representation (LSTUR). We aim to enhance these models to predict user engagement effectively. By integrating advanced features, optimizing performance metrics, and focusing on beyond-accuracy objectives such as diversity, novelty, and serendipity, we strive to improve news recommendation accuracy and user experience. The project's evaluation follows stringent metrics like AUC, MRR, and nDCG@K, ensuring our solutions meet high standards. Our results will be benchmarked and submitted to the RecSys Challenge leaderboard, with detailed documentation and reproducibility ensured through comprehensive coding and reporting practices.

Question: Write also the introduction

Answer: Introduction The ACM RecSys 2024 Challenge gives us a great opportunity to work on improving news recommendation systems using the new Ekstra Bladet News Recommendation Dataset (EB-NeRD). This dataset includes user interaction logs from the Danish news site Ekstra Bladet, collected over six weeks from April 27 to June 8, 2023. It contains detailed information about the news articles such as titles, abstracts, bodies, categories, and more. Our task is to develop algorithms that can accurately predict which articles users will engage with, which is a complex and interesting problem.

Our group, WeKnowWhatYouWant, has decided to use two advanced algorithms for this challenge: the Neural Propagation Algorithm (NPA) and the Long-Short Term User Representation (LSTUR). NPA is good at capturing the hierarchical structure of news content, while LSTUR focuses on both the long-term and short-term interests of users. These algorithms are particularly suitable for this task because they can effectively handle large amounts of text data and user interaction histories.

First, we will implement the baseline model provided in the EB-NeRD repository. Then, we will work on integrating our chosen algorithms. Our approach includes a lot of data preprocessing, feature engineering, and model optimization to boost the performance of our recommendations. We will evaluate our models using common metrics like the area under the ROC curve (AUC), mean reciprocal rank (MRR), and normalized discounted cumulative gain (nDCG@K). Additionally, we will consider beyond-accuracy objectives such as diversity, serendipity, novelty, and coverage to make sure our recommendations are not only accurate but also diverse and engaging.

By participating in this challenge, we hope to improve news recommendation systems and gain a deeper understanding of how users interact with digital news. We will document all our findings and methods and submit our results to the RecSys Challenge leaderboard. This project is a great chance for us to apply what we've learned and contribute to the field of recommender systems.