NewsRecs: A Mobile App Framework for Conducting and Evaluating Online Experiments for News Recommender Systems

Noah Janzen[a] and Fatih Gedikli[b]
Institute of Computer Science, University of Applied Sciences Ruhr West, Mülheim an der Ruhr, Germany

Keywords: News Recommender Systems, Online Evaluation, News App, Mobile.

Abstract: News recommender systems are ubiquitous on the web. Intensive research has been conducted over the last decades, resulting in the continuous proposal of new recommendation techniques based on Machine Learning models. To evaluate the performance of recommendation algorithms, offline experiments, user studies, and online experiments should ideally be carried out one after the other so that the candidates move through a quality funnel. However, our literature review of multiple academic papers shows that new models have generally been evaluated using offline experiments only. Presumably, this is because researchers rarely have access to a production system. This work attempts to alleviate this problem by presenting a framework that can be used to evaluate recommendation models for news articles in an online scenario. The framework consists of a mobile app in which users can receive recommendations from different algorithms depending on their assigned group and rate them in multiple ways. The backend collects log data and makes it available for the final evaluation. The specific contributions our article will make are as follows: (1) A thematic review of 27 academic experiments from the news recommendation domain focusing on the evaluation design. (2) An open-source mobile app framework for conducting and evaluating online experiments.

1 INTRODUCTION

In the last decade, a lot of research has been done in the field of news recommender systems, focusing on the development of new recommender models. Hence, a variety of different models are available today for different use cases. In a real-world scenario, an application designer must make a decision about the most appropriate model for the specific application in which the model is to be deployed.

The selection of a recommendation model is usually based on experiments comparing the performance of a set of models. Offline experiments utilize existing data sets and a predefined protocol simulating the user behaviour to determine the performance of a model (e.g. accuracy). A more elaborate option is a user study: A small group of potential users will be asked to perform a series of tasks with the system. Subsequently, the test persons answer questions about their experience. Lastly, experiments can also be conducted with a real system, which are referred to as online experiments. The performance of the recommendation algorithms is evaluated on the basis of real users, who are typically unaware of an experiment, which is why online experiments are seen as the most trustworthy experiments. Typically, the selection of a suitable recommendation model is a gradual process involving both offline and online experiments, as well as user studies where appropriate. First, an extensive online experiment is typically conducted to filter out unsuitable approaches (screening). A relatively small set of potentially suitable models remains, which are tested with the help of more expensive user studies or online experiments to confirm the usefulness in practice (Ricci et al., 2022).

Offline experiments are appealing because they do not require interaction with real users and allow comparison of a wide range of possible algorithms at relatively low cost. Online experiments, on the other hand, offer the advantage that their results provide the strongest evidence because the system is used by real users performing real tasks. In the real world, a recommender system typically influences the behavior of its users. This change in behavior cannot be measured using offline experiments alone because they work with synthetic or historical user interaction data, i.e., the historical user behavior will remain static. A seri-

[a] https://orcid.org/0000-0001-7066-7861
[b] https://orcid.org/0000-0001-6190-0449

Janzen, N. and Gedikli, F.
NewsRecs: A Mobile App Framework for Conducting and Evaluating Online Experiments for News Recommender Systems.
DOI: 10.5220/0011658000003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 3*, pages 267-275
ISBN: 978-989-758-623-1; ISSN: 2184-433X

ous problem for academic research is that it rarely has access to a production system that allows online experiments, and therefore often relies on offline experiments (Ricci et al., 2022; Castells and Moffat, 2022).

The following sections are structured as follows. Section 2 examines evaluation practices in academic research, analyzing 27 papers that propose novel news recommendation models. Section 3 presents the requirements and the system design, Section 4 the implementation and Section 5 the major features of the proposed framework. Finally, the main findings are concluded in Section 6.

2 RELATED WORK

News analytics plays an important role in news recommender systems (Gedikli et al., 2021), and also in communication sciences in general (Nicholls and Bright, 2019). One of the main challenges is the implementation of a long-term evaluation of a news recommender system. We analyzed various papers proposing novel news recommendation models in order to answer the following questions: How often are models evaluated using offline or online experiments in academic research? What metrics are used to evaluate the models?

2.1 Methodology

A semi-systematic approach was used to identify relevant papers on the topic of news recommenders. First, the Google Scholar search engine and a number of digital libraries[1] were queried using the search term "news recommendations". Results prior to 2017 were filtered out, as only current evaluation practice is of interest. The papers received were then examined with regard to their relevance in terms of title and abstract. To answer the above research questions, only papers that present and evaluate a model for generating recommendations for news are of interest. The remaining papers were then used as the starting point of a snowball process. Additionally, a meta-study could be found that also analyzes research for news recommendation systems (Amir et al., 2022). Finally, this served to validate our own findings.

2.2 Survey of Existing Approaches

A total of 27 research papers were examined that focus on the development and evaluation of novel news recommendation models. The eldest paper dates from 2017, the most recent from 2022, and the majority of research papers were published in 2019 or later (85%).

Only 5 papers evaluated the developed model in an online experiment. In 22 research papers, the model was evaluated using only offline experiments. All 5 research papers that examined their model with online experiments also set up offline experiments in advance. This confirms our assumption from the beginning that academic research often relies only on offline experiments (Castells and Moffat, 2022). The reason for this is assumed to be that they rarely have access to a productive system that allows online experiments.

2.3 Evaluation Metrics

On average, the proposed models were evaluated using 3.7 evaluation metrics (median: 3). All papers used at least two metrics. 70% of all models were evaluated with 2 or 3 metrics. Research papers that evaluated both offline and online tended to use more metrics (5, 5, 7, 7, 12 metrics).

Table 1 shows all research papers examined with the evaluation metrics used in each case. The last row indicates in how many papers the respective metric was used. Metrics that were used less than twice are not shown for clarity. It can be seen that the Normalized Discounted Cumulative Gain (nDCG), Area Under the ROC Curve (AUC), and Mean Reciprocal Rank (MRR) metrics are by far the most frequently used for offline evaluation of news recommendation systems (18, 17, and 15 times, respectively). This is likely due to the fact that news recommendations are typically displayed in the form of a list and these metrics are used to evaluate recommendation lists. This is followed far behind by F1 score (F1), hit rate (also hit ratio), click through rate (CTR) and precision (6, 5 and 4 uses).

Since metrics for online evaluation are particularly relevant to the app framework, Table 2 includes only the research papers that conducted online experiments. The table shows all metrics that were used in the online experiments of the papers.

By far the most popular is the Click Through Rate (CTR) used for online evaluation. The CTR divides the number of clicks on a recommendation by the number of views of that recommendation. The second most used metric is the number of clicks per session. Session is defined here as the use of the recommendation service by a user. If no action is performed within a defined period of time (e.g. 15 minutes), the session ends. Except for the metrics CTR and Clicks, there

[1] Springer Link, ACM Digital Library, arXiv.org and ResearchGate were reviewed.

Table 1: Evaluation metrics used in the experiments.

	Experiment	nDCG	AUC	MRR	F1	Hit rate	CTR	Precision	Recall	ILD	Clicks
1	DRN (Zheng et al., 2018)	•					•				
2	RAKGG (Liu et al., 2021)	•				•		•	•		
3	OF-SMR (Mulder et al., 2021)						•			•	
4	PLM-NR (Wu et al., 2021)	•	•	•							•
5	RWTHIN (Symeonidis et al., 2020)	•						•		•	
6	NPA (Wu et al., 2019b)	•	•	•							
7	D-HAN (Zhao et al., 2021)	•				•					
8	PTG (Koo et al., 2020)			•		•					
9	GERL (Ge et al., 2020)	•	•	•							
10	UPDNN-TNR (Hu et al., 2020a)		•		•						
11	CSRN (Bai et al., 2020)			•		•					
12	ENR-MU (Okura et al., 2017)	•	•	•			•				•
13	DNN-NR (Park et al., 2017)			•				•	•		
14	NRMS (Wu et al., 2019c)	•	•	•							
15	NRAML (Wu et al., 2019a)	•	•	•							
16	NRLSUR (An et al., 2019)	•	•	•							
17	PNRML (Qi et al., 2020)	•	•	•							
18	AMM (Zhang et al., 2021)	•	•	•							
19	JKP-RGC (Tian et al., 2021)	•	•	•							
20	DKN (Wang et al., 2018)		•		•						
21	FIM (Wang et al., 2020)	•	•	•							
22	GNNR-UPD (Hu et al., 2020b)		•		•						
23	KRED (Liu et al., 2020)	•	•		•	•					
24	ASA-IPNR (Yoneda et al., 2019)	•					•				
25	GNNR-UEPIM (Qiu et al., 2022)	•	•	•							
26	EHUP-KG (Wang et al., 2019)				•			•	•		
27	TEKG (Lee et al., 2020)		•		•						
Frequency		18	17	15	6	5	4	4	3	2	2

• Metric used in an offline experiment • Metric used in an online experiment

are no common metrics that have been used for evaluation in several research papers (see Table 2). This could be due to differences in the production systems with which the experiments were conducted.

In one paper, an online experiment was conducted on the Dutch news platform Blendle (Mulder et al., 2021). On the platform, after reading a news article, users can mark the article as a favorite. The researchers defined the favorite ratio as the number of users per user group (A or B) who marked an article as a favorite, divided by the number of users in the same group who finished reading the article. The researchers assume that the ratio is an indicator of user satisfaction with the article. Such a metric is highly dependent on the productive system under which the experiments are performed. Consequently, if the Favorites feature does not exist in the productive system, the Favourite Ratio cannot be determined either.

Despite the differences in the use of the various metrics, many metrics have in common that they are based on clicks on a recommendation (CTR, Clicks/Session, Precision@k, nDCG, CTR per article, CTR per recommendation set, Clicks, Click Users/Session). An essential requirement of the app framework can be derived from this: Clicks on a recommendation must be logged.

3 REQUIREMENTS AND SYSTEM DESIGN

Next, we introduce our NewsRecs framework, which allows researchers to compare several recommenders

Table 2: Evaluation metrics used in the online experiments.

Experiment	CTR	Clicks/Session	Precision@k	nDCG	ILS	CTR per article	CTR per rec. set	Compl. rate of rec.	Favourite ratio	Presentation charac.	Source diversity	Clicks	Pageviews	Sessions	Duration	Click Users/Session
DRN (Zheng et al., 2018)	•		•	•	•											
OF-SMR (Mulder et al., 2021)	•					•	•	•	•	•	•					
PLM-NR (Wu et al., 2021)												•	•			
ENR-MU (Okura et al., 2017)	•	•												•	•	
IPNR (Yoneda et al., 2019)	•	•														•
Frequency	4	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1

in an online experimental setting. Since the global web traffic on mobile devices has surpassed desktop long ago, we developed a mobile app for mobile devices and therefore focus on mobile users. However, the frontend can easily be converted into a web application because the UI-framework we use is based on web technologies.

3.1 Framework Requirements

The target users can be divided into two groups: On the one hand, there are app users who use the mobile app to receive news recommendations. On the other hand, researchers use the software to conduct online experiments, to create user surveys, and to evaluate new algorithms or potentially user interfaces.

In order to be able to associate data with users, a registration step is required. Therefore, an app user can create an account and log in. He can also edit his user profile within the app. The core feature of the app is to receive news recommendations in a list. A user can press on a news recommendation so that the entire article opens within an in-app browser for reading. After reading the article, the user can rate the recommendation. On the other side, researchers are able to design A/B-tests and create surveys for app users. Researchers can also download all log data to evaluate the alternatives tested.

3.2 Software Architecture

The NewsRecs framework implements a client-server architecture. Users consume news recommendations from a central server on their mobile client (see Figure 1). An external news server must provide metadata of news articles via a GraphQL or REST interface, such as the titles, publication dates and the links of the news articles. The framework then takes care of the rest. The news server and news database are not part of the NewsRecs framework, but can be easily implemented by using, e.g., one of many Python-libraries for scraping news articles.

The NewsRecs server accesses the news server to load news items. The recommendation algorithms to be evaluated must be implemented in the NewsRecs server to generate recommendations to users. The NewsRecs server is connected to a database that stores user, usage, and survey data. Users access the NewsRecs server via a REST interface, for example, to register, receive news recommendations, or rate recommendations on a Likert scale designed by the researcher. Researchers can directly access the NewsRecs database to download user usage data and survey responses. Researchers can also create new surveys on the NewsRecs server. Online experiments can run over several days, weeks or months, making long-term studies possible, which is one of the biggest challenges in news recommender systems.

3.3 Technology Stack

NewsRecs uses the established MERN stack, consisting of **M**ongoDB as a document-oriented NoSQL database, **E**xpress.js as a server-side web framework, **R**eactNative as a UI framework for cross-platform development of native apps for the Android and iOS operating systems, and **N**ode.js as a server-side JavaScript runtime environment. NestJS has been added to this stack. NestJS is a framework to develop scalable server applications and provides an abstraction layer above the common Node.js frameworks. Both the frontend and the backend are written in the TypeScript programming language.

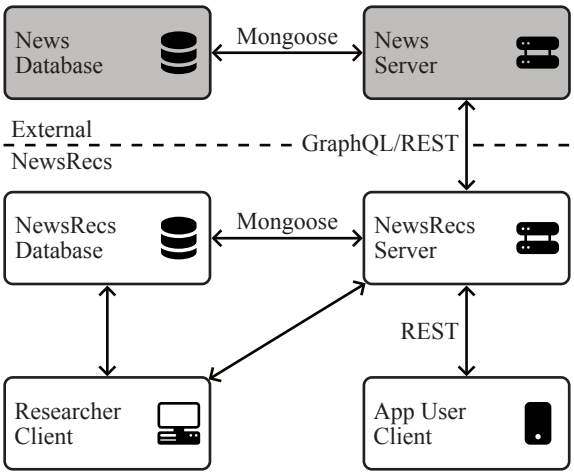


Figure 1: Architecture of the NewsRecs Framework.

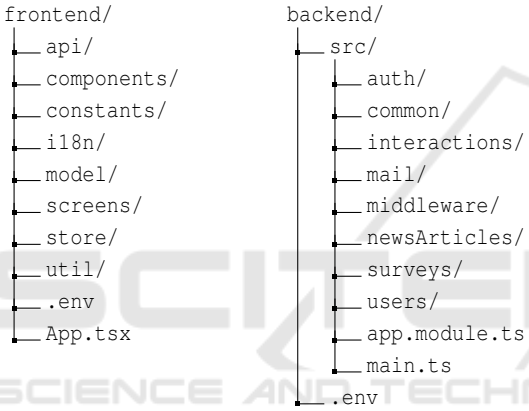


Figure 2: File structure of the NewsRecs framework.

4 IMPLEMENTATION

NewsRecs code and examples are MIT-licensed to minimize legal impediments to adoption. The code is available on GitHub[2].

4.1 Frontend

The mobile app forms the front end, which was implemented using the React Native framework. Figure 2 shows the relevant folders and files in the root directory of the frontend, each of which is briefly discussed below:

- In the `api` folder there are TypeScript modules, that provide functions for communicating with the backend.
- All reusable React components are stored in the `components` directory.
- The `constants` folder contains modules with general JavaScript objects like the app's color scheme.
- The mobile app currently supports the languages German and English. Depending on the system language of the smartphone, the language of the app is automatically adjusted. Localization files are stored in the `i18n` directory.
- The different screens of the app are located in the `screens` folder.
- The widely used Redux library is used to manage state information within the mobile app. The Redux configuration is performed in the `store` directory.
- The `util` folder contains modules with helper functions that are used in various React components.
- The environment variables of the app are stored in an `.env` text file.
- The starting point of the app is the `App.tsx` file. This is where the navigation logic is located, implemented using the React Navigation library.

4.2 Backend

The file structure of the backend is described below (see Figure 2).

- The `auth` directory contains the application's authentication logic. For authentication, the OAuth 2.0 protocol was implemented, which is an industry standard for authentication.
- The `interactions` folder contains the module, which manages interactions between an user and a news article.
- The `mail` directory contains the module responsible for sending e-mails which is, e.g., required for user registration or password reset.
- The `newsArticles` folder contains the module which provides an interface to load news recommendations. News metadata is first loaded from the news server. Depending on his group membership, a user receives news recommendations provided by different algorithms. The membership of a group depends on the ID of the user; thus, it is de facto randomized. At this point, when used in research, the recommendation model to be tested and a baseline model must be implemented for comparison. In this way, A/B-tests can be carried out.

[2]https://github.com/noah-janzen/news-recs

- The `surveys` directory contains a module, which provides an interface to create surveys and to query and store survey responses.
- The `users` directory contains a module, which provides an interface for retrieving and editing the user data of the logged-in user.
- The `app.module.ts` file contains the root module of the Nest application.
- The `main.ts` file is the initial file of the application that creates the nest application instance.
- As in the frontend, there is an `.env` file in the backend that contains all environment variables.

5 MAJOR FEATURES

This section briefly explains the main use cases of the framework.

5.1 Create User Surveys

A core feature of the application is the creation of surveys that users can answer in the app. This allows quantitative and qualitative questions to be answered. Surveys are defined by text files in JSON format in the backend. A researcher can create a new JSON file in the `surveys/data/` path for this purpose. The basic structure of this file can be seen in Listing 1. The `surveyId` must be an unique positive integer number. The `startDate` property specifies the time when the survey is enabled for answering. Similarly, the `endDate` property determines until when the survey can be answered. The `startsAfterDaysSinceRegistration` property specifies after how many days since registration an user can view and answer the survey, since users should use the app for a certain period of time before being surveyed. The `durationInDays` property contains the time period in days that the user has to answer the questions. The `questions` property is an array that contains the questions.

Listing 1: Definition of a survey.

```
{
  "surveyId": 0,
  "startDate": "2022-07-20T10:00:00.000+00:00",
  "endDate": "2022-08-27T10:00:00.000+00:00",
  "startsAfterDaysSinceRegistration": 3,
  "durationInDays": 7,
  "questions": [
    // Question Objects
  ]
}
```

Listing 2 shows the definition of a single question object in the context of a concrete survey. It contains the question in the supported languages of the app. Currently, three different question type options are implemented (see Figure 3). Depending on the question type, the property `answerOptions` must be defined. This property is an array of multiple `answerOption` objects. Such an object is identified by an integer, non-negative value and the translations for the answer.

Listing 2: Definition of a `question` object.

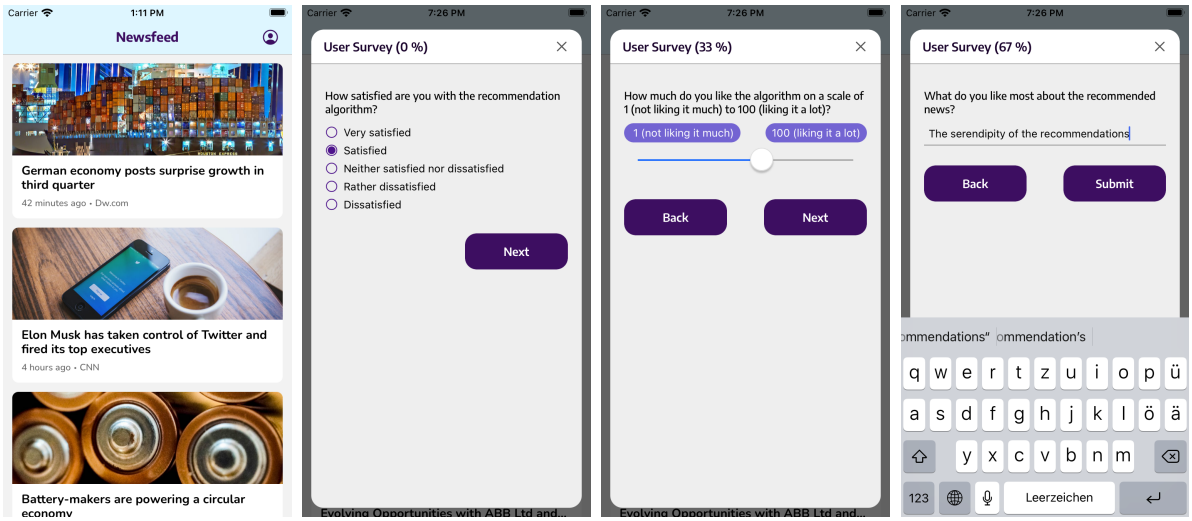
```
{
  "question": {
    "en": "How satisfied are you with the recommendation algorithm?",
    "de": "Wie zufrieden bist Du mit dem Empfehlungsalgorithmus?"
  },
  "questionType": "RADIO",
  "answerOptions": [
    {
      "value": 4,
      "answer": {
        "en": "Very satisfied",
        "de": "Sehr zufrieden"
      }
    },
    // More answer options
  ]
}
```

In the backend database, survey responses are stored as individual documents. Users are able to change previous answers while conducting a survey. Once the user has completed the survey, no more answers can be edited. Each document includes a timestamp indicating when the question was first answered, the user id, the survey id, the question id, the question type, the given answer and an optional timestamp recording when the answer was last modified.

5.2 Evaluate Interaction Data

An interaction is a link between an user and a news item that has been suggested to him. A distinction is made between the following three types of interaction:

(a) In the newsfeed, a news article is displayed to an user in the visible area (viewport).



(a) Newsfeed. (b) Question type `RADIO`. (c) Question type `RANGE`. (d) Question type `TEXT`.

Figure 3: Screenshots of the mobile app. Subfigures (b), (c) and (d) show the different question types available for surveys.

(b) As in (a), but here the user presses on the news article to read it.

(c) After reading a news article, a modal dialog opens where the user can rate the recommendation. Rating a news recommendation is the third interaction type.

A separate document is created in the database for each step. For example, if a user sees a recommendation in the newsfeed, reads the corresponding article and then rates it, three interactions are stored in the database.

Three different input elements are available for rating a recommendation (a binary input element, a range input element and a free-text field). Before deploying the app, a researcher has to decide between one of these three input element types, which must be specified in the `.env` file in the frontend. The structure of a document depends on the interaction type. All interaction documents contain the user ID, the message article ID, and a timestamp. Interactions of type (a) and (b) additionally store the flag `clicked`, which indicates whether the user pressed on the news article to read it. Interactions of type (c), on the other hand, additionally store the input element type (`binary`, `range`, or `text`) and the actual rating (a string or a numeric value). The interactions can be loaded via direct database access and analyzed locally on a researcher's computer, for example.

6 CONCLUSIONS

Newly proposed recommendation models need to be evaluated to assess their effectiveness from the user's perspective. For this purpose, offline experiments, user studies and online experiments are conducted gradually. Especially in academic research, there is rarely an opportunity to implement online experiments because it usually does not have access to a production system. To solve this problem, we developed a software framework that allows researchers to conduct online experiments by using a mobile news app with a recommendation engine.

First, 27 recent and related papers proposing new recommendation techniques for news articles were examined. In the majority of the research (22 papers, 81%), only offline experiments were conducted, and no further user studies or online experiments were performed. The most popular metrics for online evaluation are click-based metrics.

In general, the metrics used are highly dependent on the production system used to implement the experiment.

In a second step, a mobile app was developed that displays news recommendations to users. Users are divided randomly into two groups, each of which receives news recommendations from a different algorithm. User interactions are recorded and stored in a database. In addition, researchers can create surveys that are displayed to users in the app.

We also conducted a usability study with three test users to obtain feedback for future usability improvements. All users had a positive impression of the app

and made some suggestions for improvement. For example, it is important for the users that they receive relevant recommendations in their native language.

With NewsRecs we want to contribute to the dissemination of online experiments for news recommender in order to obtain more trustworthy results in future work. We welcome the community to contribute on GitHub and provide feedback in order to jointly set the direction for future developments.

REFERENCES

Amir, N., Jabeen, F., Ali, Z., Ullah, I., Jan, A., and Kefalas, P. (2022). On the current state of deep learning for news recommendation. *Artificial Intelligence Review*.

An, M., Wu, F., Wu, C., Zhang, K., Liu, Z., and Xie, X. (2019). Neural news recommendation with long- and short-term user representations. In *ACL 2019*, pages 336–345. ACL.

Bai, B., Zhang, G., Lin, Y., Li, H., Bai, K., and Luo, B. (2020). CSRN: Collaborative sequential recommendation networks for news retrieval. *CoRR*, abs/2004.04816.

Castells, P. and Moffat, A. (2022). Offline recommender system evaluation: Challenges and new directions. *AI Magazine*, 43(2):225–238.

Ge, S., Wu, C., Wu, F., Qi, T., and Huang, Y. (2020). Graph enhanced representation learning for news recommendation. In *WWW 2020*, pages 2863–2869. ACM.

Gedikli, F., Stockem Novo, A., and Jannach, D. (2021). Semi-automated identification of news story chains: A new dataset and entity-based labeling method. In *INRA 2021*, pages 29–42. CEUR-WS.org.

Hu, L., Li, C., Shi, C., Yang, C., and Shao, C. (2020a). Graph neural news recommendation with long-term and short-term interest modeling. *Inf. Process. Manage.*, 57(2).

Hu, L., Xu, S., Li, C., Yang, C., Shi, C., Duan, N., Xie, X., and Zhou, M. (2020b). Graph neural news recommendation with unsupervised preference disentanglement. In *ACL 2020*, pages 4255–4264. ACL.

Koo, B., Jeon, H., and Kang, U. (2020). Accurate news recommendation coalescing personal and global temporal preferences. In *PAKDD 2020*, pages 78–90. Springer.

Lee, D., Oh, B., Seo, S., and Lee, K.-H. (2020). News recommendation with topic-enriched knowledge graphs. In *CIKM 2020*, pages 695–704. ACM.

Liu, D., Lian, J., Liu, Z., Wang, X., Sun, G., and Xie, X. (2021). Reinforced anchor knowledge graph generation for news recommendation reasoning. In *KDD 2021*, pages 1055–1065. ACM.

Liu, D., Lian, J., Wang, S., Qiao, Y., Chen, J.-H., Sun, G., and Xie, X. (2020). KRED: Knowledge-aware document representation for news recommendations. In *RecSys 2020*, pages 200–209. ACM.

Mulder, M., Inel, O., Oosterman, J., and Tintarev, N. (2021). Operationalizing framing to support multiperspective recommendations of opinion pieces. In *FAccT 2021*, pages 478–488. ACM.

Nicholls, T. and Bright, J. (2019). Understanding news story chains using information retrieval and network clustering techniques. *Communication Methods and Measures*, 13(1):43–59.

Okura, S., Tagami, Y., Ono, S., and Tajima, A. (2017). Embedding-based news recommendation for millions of users. In *KDD 2017*, pages 1933–1942. ACM.

Park, K., Lee, J., and Choi, J. (2017). Deep neural networks for news recommendations. In *CIKM 2017*, pages 2255–2258. ACM.

Qi, T., Wu, F., Wu, C., Huang, Y., and Xie, X. (2020). Privacy-preserving news recommendation model learning. In *EMNLP 2020*, pages 1423–1432. ACL.

Qiu, Z., Hu, Y., and Wu, X. (2022). Graph neural news recommendation with user existing and potential interest modeling. *ACM Trans. Knowl. Discov. Data*, 16(5).

Ricci, F., Rokach, L., and Shapira, B., editors (2022). *Recommender systems handbook*. Springer, third edition.

Symeonidis, P., Kirjackaja, L., and Zanker, M. (2020). Session-aware news recommendations using random walks on time-evolving heterogeneous information networks. *User Modeling and User-Adapted Interaction*, 30:1–29.

Tian, Y., Yang, Y., Ren, X., Wang, P., Wu, F., Wang, Q., and Li, C. (2021). Joint knowledge pruning and recurrent graph convolution for news recommendation. In *SIGIR 2021*, pages 51–60. ACM.

Wang, H., Wu, F., Liu, Z., and Xie, X. (2020). Fine-grained interest matching for neural news recommendation. In *ACL 2020*, pages 836–845. ACL.

Wang, H., Zhang, F., Wang, J., Zhao, M., Li, W., Xie, X., and Guo, M. (2019). Exploring high-order user preference on the knowledge graph for recommender systems. *ACM Trans. Inf. Syst.*, 37(3).

Wang, H., Zhang, F., Xie, X., and Guo, M. (2018). DKN: Deep knowledge-aware network for news recommendation. In *WWW 2018*, pages 1835–1844. IW3C2.

Wu, C., Wu, F., An, M., Huang, J., Huang, Y., and Xie, X. (2019a). Neural news recommendation with attentive multi-view learning. In *IJCAI 2019*, pages 3863–3869. AAAI Press.

Wu, C., Wu, F., An, M., Huang, J., Huang, Y., and Xie, X. (2019b). NPA: Neural news recommendation with personalized attention. In *KDD 2019*, pages 2576–2584. ACM.

Wu, C., Wu, F., Ge, S., Qi, T., Huang, Y., and Xie, X. (2019c). Neural news recommendation with multi-head self-attention. In *EMNLP-IJCNLP 2019*, pages 6389–6394. ACL.

Wu, C., Wu, F., Qi, T., and Huang, Y. (2021). Empowering news recommendation with pre-trained language models. In *SIGIR 2021*, pages 1652–1656. ACM.

Yoneda, T., Kozawa, S., Osone, K., Koide, Y., Abe, Y., and Seki, Y. (2019). Algorithms and system architecture

for immediate personalized news recommendations. In *WI 2019*, pages 124–131. ACM.

Zhang, Q., Jia, Q., Wang, C., Li, J., Wang, Z., and He, X. (2021). AMM: Attentive multi-field matching for news recommendation. In *SIGIR 2021*, pages 1588–1592. ACM.

Zhao, Q., Chen, X., Zhang, H., and Ma, S. (2021). D-HAN: Dynamic news recommendation with hierarchical attention network. *CoRR*, abs/2112.10085.

Zheng, G., Zhang, F., Zheng, Z., Xiang, Y., Yuan, N. J., Xie, X., and Li, Z. (2018). DRN: A deep reinforcement learning framework for news recommendation. In *WWW 2018*, pages 167–176. ACM.