
Two-Stage News Recommendation via Candidate Generation and Ranking Models

Mahnoor Sheikh

1. Abstract

The rapid growth of online news has made personalised recommendation essential for helping users efficiently find relevant information. This makes recommendation systems the backbone of successful social recommendation platforms. However, the quality of recommendations heavily depends on effective ranking of candidate articles. Modern platforms increasingly rely on two-stage retrieval architectures, where a large pool of candidate articles is first retrieved and then ranked using personalised features. This project develops and evaluates such a system using the MIND-small dataset.

After cleaning and preprocessing news content and user behaviour logs, a retrieval model combining TF-IDF content similarity, CTR-based popularity priors, and user category preferences is used to construct a candidate pool. Multiple ranking models (logistic regression, XGBoost, and a hybrid MLP integrating TF-IDF (SVD) and DistilBERT embeddings) are then trained to order these candidates. Results demonstrate that while traditional models capture global patterns reasonably well, the proposed MLP reranker achieves huge gains in personalised ranking, significantly improving Hit@5, MRR@5, and nDCG@5 performance.

2. Introduction

Personalised news recommendation has become imperative to digital platforms, where users expect timely, relevant, and diverse content tailored to their interests.

Prior works have explored multiple ways to perfect this. Li and Li (2011) introduced SCENE, a two-stage news recommendation architecture that retrieves a candidate pool using topic modelling and adaptive user profiling, and then applies a personalised re-ranking step based on contextual multi-armed bandits. Their results showed huge improvements in click-through rate and efficiency over single-stage recommenders, demonstrating the effectiveness of separating retrieval from ranking. Qi et al. (2021) improved recommendation accuracy by proposed PP-Rec, which models both personalised user interest and time-aware news popularity, through an attention mechanism. Their approach achieved higher AUC and nDCG than content-only or popularity-only baselines,

showing the significance of integrating content relevance and CTR signals. Wu et al. (2022) addressed fairness concerns in re-ranking by introducing FairRank, a neural ranking framework. Their results showed that fairness constraints can reduce exposure bias across publishers with minimal performance downgrade.

These studies align with the structure of my approach. Following the industry-standard procedure, this project first generates a candidate pool using content similarity and behavioural priors, and then applies progressively stronger ranking models to refine personalised recommendations. The objective is to evaluate how classical, tree-based, and deep learning models differ in their ability to capture user preferences within this two-stage framework.

3. Problem Description

Online news platforms must efficiently recommend articles that match a user's evolving interests, despite large volumes of content available on the internet and sparse interaction signals associated with it.

In this project, the goal is to design a system that, for each impression of a user upon log-in, recommends five relevant news articles based on their historical click behaviour. The setting follows a two-stage retrieval architecture: a broad candidate pool is first generated using TF-IDF similarity, popularity priors derived from click-through rates, and user-specific category preferences. More specifically, each candidate article is scored using content similarity, news popularity, category preference, and source familiarity, and the top K articles make up the candidate pool. A ranking stage then orders these candidates using machine-learning models trained on impression-level data. The central challenge lies in accurately capturing user preferences under sparse feedback, balancing semantic relevance with behavioural patterns, and ensuring high top-K ranking quality. This set-up reflects the structure followed by modern industrial news recommendation systems, where effective retrieval and personalised reranking jointly determine overall recommendation quality and output.

4. Data

MIND: Microsoft News Dataset is a large-scale dataset for news recommendation research, collected from anonymised behaviour logs of the [Microsoft News](#) website. It contains about 160k English news articles and more than 15 million impression logs generated by 1 million users.

Here, the training and validation sets of MIND-small have been used, which are randomly sampled-down versions of MIND. Both combined contain 65,238 English news articles and 230,117 impression logs generated by 94,057 users (50,000 users each in both datasets).

Features of the 'news' dataset are:

```
"news_id": unique identifier of news item
"category": category news belongs to
"subcategory": subcategory news belongs to
"title": title of news item
"abstract": abstract body of news item
"url": web address of news item
"title_entities": JSON-formatted list of
  title entity objects
"abstract_entities": JSON-formatted list of
  abstract entity objects
```

Features of the 'behaviors' dataset are:

```
"impression_id": unique identifier of
  impression log/event, i.e. multiple
  articles are recommended to a user upon
  log-in
"user_id": user whose impression it is
"time": date and time of this impression
"clicked_news": historical news click
  behaviours (news_id) of this user
  before this impression
"impressions": clicked and non-clicked
  events (news_id with label '1' for
  clicked and '0' for non-clicked)
```

5. Methodology

5.1. Data Cleaning and Preprocessing

To make the raw datasets usable for recommendation modeling, a preprocessing pipeline was applied separately to the news metadata and user behaviour tables, followed by the computation of click-through rate (CTR) priors.

5.1.1. NEWS METADATA PREPROCESSING

The training and validation sets together contained 65,238 unique news articles, with missing values in the `abstract` field (2,666 in train and 2,021 in validation).

- For the `title` and `abstract` fields, HTML tags and character entities were stripped with *BeautifulSoup*,

rule lines were removed using regular expressions, uni-code characters and punctuation were standardised, and excessive whitespace or line breaks were collapsed to produce sentences suitable for tokenisation.

- For articles with extremely short or missing abstracts (fewer than 30 characters), only the cleaned title was used. Otherwise, cleaned titles and abstracts were concatenated to form a `content` field for bag-of-words models.
- Articles were marked as `is_short` if their `content` length was below 50 characters or contained fewer than five tokens. Only 829 of 51,282 training articles were flagged as short.
- Source (publisher) information was reconstructed from the URL and textual hints. Most items were mapped to the MSN portal domain (`assets.msn.com`), but prominent outlets like AP, BBC, etc, were also recovered.

5.1.2. USER BEHAVIOUR PREPROCESSING

The raw behaviour tables are converted from a compact string format of impressions and clicks into a univariate representation of user-item interactions.

- The `impressions` field is split by exploding the candidate list to produce one row per (`impression_id`, `user_id`, `time`, `news_id`, `label`), where `label=1` indicates a clicked and `label=0` a non-clicked recommendation.
- The impression table was merged with the news metadata on `news_id`, attaching `category_merged` and `source_key` to each impression.
- Three CTR statistics were computed: Per-news CTR (popularity of individual articles), Per-category CTR (how often each category is clicked), and Per-source CTR (click propensity for each publisher). To avoid instability for rarely shown items, Laplace smoothing $\alpha = 1$ was applied. High engagement categories and sources via smoothed CTR metrics in the training data are displayed in [Figures 1](#) and [2](#).
- For modelling personalised preferences, a user-level click history was built by restricting the impression table to rows with `label=1`, sorting by timestamp for ordered clicked `news_ids`.

Applying this pipeline to the training and validation datasets resulted in 50,000 users in each split, with 5,943 users overlapping between train and validation sets.

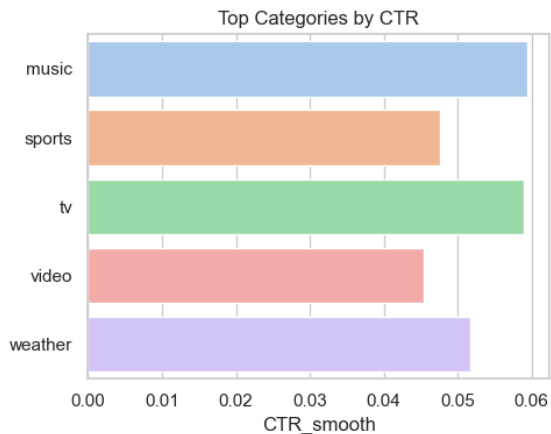


Figure 1. Top categories by CTR in the training data.

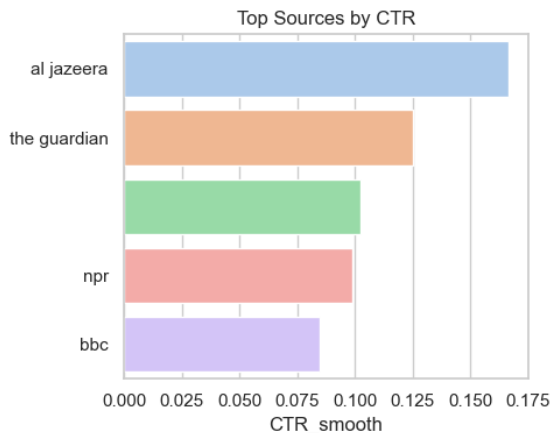


Figure 2. Top sources by CTR in the training data.

5.2. Candidate Generation: Recall@K

The recommendation pipeline followed a two-stage design: candidate generation first narrowed the full news catalogue down to a small, high-probability set of candidate articles, after which ranking models (Section 5.3) learned to rank these candidates. This section describes the candidate generator and how the pool size K was selected using Recall@K on validation impressions. This approach balances relevance with computation cost.

5.2.1. SETUP: TF-IDF CONTENT SIMILARITY

To model proximity between articles’ content, the cleaned content field was converted into TF-IDF vectors using `TfidfVectorizer` with the following parameters:

- minimum document frequency = 3,
- maximum document frequency = 0.8,

- n-grams = unigrams and bigrams, and
- L2 normalisation and lowercase unicode text.

The vectoriser was fitted on the training news dataset and applied to both train and validation articles, outputting a feature space of 140,562 dimensions. For each clicked article, a cached similarity lookup retrieved the top 300 most similar validation articles using cosine similarity.

For each user, the last $k=5$ clicked articles from both the training and validation history were considered. For each of these clicks, its top-300 neighbours were scored with a decaying weight of $1/(\text{similarity rank} + 5)$. The contributions from all recent clicks were summed, producing a content similarity score for each candidate news item. This procedure values articles that are near multiple items in the user’s recent clicked history, rather than on a single nearest neighbour.

5.2.2. SETUP: POPULARITY AND PREFERENCE PRIORS

To capture global popularity and user-specific preferences, three additional signals were incorporated.

- **News-level popularity prior:** From the expanded impression table, per-news CTR was computed with Laplace smoothing. For each `news_id`, the smoothed CTR was

$$\text{ctr_smooth} = \frac{\text{clicks} + 1}{\text{shown} + 2}.$$

This value was stored as `prior_news_ctr`. When a validation article had no direct prior, its score fell back to its category prior and then to its source prior and as a final resort, to a global CTR baseline.

- **User category preference:** For each user, all clicked training and past validation articles were mapped to their category values. Category counts were smoothed with $\alpha = 1$ and converted into probabilities via a temperature-scaled softmax (temperature $\tau = 0.8$). This gives a personalised distribution over categories, assigning higher probability to categories that the user clicks more frequently. If a user had no history, the global category distribution derived from training CTRs was used instead.
- **Source familiarity bonus:** A small binary bonus encoded whether the candidate article belonged to a news source the user had previously clicked. If the article’s source appeared in the user’s click history, a fixed bonus term ‘1’ was added; otherwise, it was zero. This encourages recommending articles from sources that the user already appears to trust, while still allowing diverse recommendations.

These priors were precomputed from the training data and stored in lightweight lookup dictionaries for efficient use during candidate generation over the large dataset.

5.2.3. SCORING SETUP: CANDIDATE FILTERING

For each validation impression, candidate articles were generated as follows:

1. **Click history construction:** The user’s training click history was combined with all validation clicks strictly earlier than the current impression timestamp to ensure a realistic recommendation protocol without any data leakage between train and validation sets.
2. **Exclusion rules:** Articles that had already been clicked by the user, as well as those flagged as `is_short`, were excluded from the pool to avoid recommending trivial or previously consumed content.
3. **Score computation:** For every remaining validation article, four components were retrieved:
 - content similarity: the aggregated content similarity score from recent clicks;
 - news popularity: the smoothed news-level CTR prior;
 - category preference: the user’s category preference probability for that article’s category (or the global category prior if the user was new);
 - source familiarity: binary, resulting in a small constant (0.05) if the article’s source had appeared in the user’s history, 0 otherwise.

These were combined into a single candidate score:

$$\begin{aligned} \text{candidate score} = & 0.65 * \text{content similarity} \\ & + 0.25 * \text{news popularity} \\ & + 0.10 * \text{category preference} \\ & + 0.05 * \text{source familiarity} \end{aligned}$$

The weights were chosen using domain knowledge and iterative experimentation to balance personalised relevance, global trends, and user-specific preferences.

4. **Top-K selection.** All scored articles were sorted in descending order, and the top K items were retained as the candidate pool for that impression.

This setup gives a smaller set of candidates that are both relevant to the user’s interests and aligned with overall platform popularity.

5.2.4. IMPLEMENTATION: SELECTING K USING RECALL@K

To choose an appropriate candidate pool size K, the generator was evaluated on a 5000-impressions random subset of validation impressions using Recall@K, where

$$\text{Recall@K} = \frac{\# \text{ impressions where clicked is in top-}K}{\# \text{ impressions with a click}}.$$

K values between 50 and 800 (step 50) were evaluated, and the impressions were processed in chronological order per user. At each impression, the candidate pool was produced as described above, and Recall@K was updated. Although a small fraction of impressions was sampled for efficiency, all users were included.

The resulting curve, which can be seen in Figure 3, shows a steady increase in recall as K grows: Recall@50 \approx 0.18, Recall@200 \approx 0.35, Recall@400 \approx 0.49, and Recall@800 \approx 0.58. Recall monotonically increases with K; however, incremental gains fall sharply beyond K=500, as the improvement from 450 to 500 is less than 0.01, and higher values of K mostly only add to the computational cost without substantial improvement in the score.

Based on this trade-off, K=500 was selected as the number of candidates generated for each user at each impression for the two-stage system. At this setting, approximately 51.6% of validation impressions had their clicked article present in the candidate pool, providing a strong foundation for the ranking models to focus on focused ordering rather than coarse retrieval.

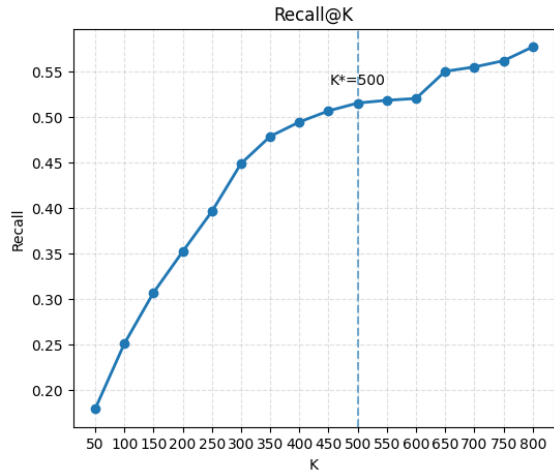


Figure 3. Recall@K evaluated for K values between 50 to 800.

5.2.5. IMPLEMENTATION: CANDIDATE POOL GENERATION ON TEST DATASET

The candidate generator was run with $K=500$ on 10,000 validation impressions to build a dataset for downstream ranking. The complete validation dataset with 73,152 impressions was not used due to computational constraints. For each impression, the following fields were stored: `impression_id`, `user_id`, `time`, `pool` (the ordered list of up to 500 candidates), `news_ids`, and `clicked` (the ground-truth clicked `news_id` (if any) for that impression).

These records were saved as a Parquet file and form the input to the ranking models described in Section 5.3. This setup efficiently separates recall-oriented candidate generation from precision-oriented candidate ranking, and ensures all ranking models are evaluated on the same fixed candidate pool for fair comparison.

5.3. Ranking Models

In the second stage, the pool of up to 500 news articles per impression was ranked so that the most relevant articles appear in the top positions. I also evaluated the same models in the standard one-stage setting used in the MIND competition, where the model ranks only the original impression list (the 20–50 articles shown to the user) without using my candidate-generation pool.

For training, I used the expanded training impressions table, which contained one row per (`user_id`, `impression_id`, `news_id`, `label`). Each row was mapped to the corresponding TF-IDF or embedding index of the article, producing:

- X_{train} : TF-IDF features for 5.84M user–article pairs,
- y_{train} : binary click labels.

For two-stage validation, the pool of 500 `news_ids` was expanded into a table with one row per candidate and label. This gave 5M candidate pairs across 10,000 sampled impressions. For one-stage validation, I used the full validation impression table, which included all articles that were originally shown to the user.

Evaluation was performed at the impression level using Hit@K, MRR@K, and nDCG@K, as defined later in Section 5.4.

5.3.1. BASELINE: LOGISTIC REGRESSION ON TF-IDF

The linear logistic regression model was trained on high-dimensional TF-IDF features. Each training pair used the 140,562-dimensional content TF-IDF vector

of the corresponding article. The model used L2 regularisation, inverse regularisation strength $C=1.0$, and `max_iter = 1000`.

The model output a click probability $p(y = 1 \mid \text{user}, \text{news})$ for each pair. For each impression, all candidates or articles were scored and sorted by the predicted probability. This logistic regression baseline serves as the simplest content-based ranking model and provides a reference point for more complex architectures.

5.3.2. SCALED-UP TREE-BASED MODEL: XGBOOST

To capture non-linear relationships in the TF-IDF feature space, I trained an XGBoost gradient-boosted tree model on the same set of training pairs. Parameters chosen were 300 trees, maximum tree depth of 6, learning rate 0.1, subsampling and column subsampling at 0.8, binary logistic objective, and histogram-based tree construction.

XGBoost also produced a probability score per user–article pair. Comparing XGBoost against the linear logistic regression baseline shows how much additional information can be extracted from the same TF-IDF features using a more flexible model.

5.3.3. MULTI-LAYER PERCEPTRON WITH DISTILBERT EMBEDDINGS (MLPv1)

Dense semantic embeddings from DistilBERT were tested. All cleaned article texts were encoded with a sentence-level DistilBERT model (`distilbert-base-nli-mean-tokens`), outputting a 768-dimensional embedding per article.

To keep training manageable, all positive clicked user–news pairs were preserved, and a randomly sampled subset of negative pairs was kept so a balanced train set of 300,000 pairs was constructed. The features were standardised using `StandardScaler`. The model architecture is as follows:

- input dimension 768,
- two hidden layers: 256, 64 with ReLU activations,
- final linear layer to a single logit,
- binary cross-entropy with logits as the loss,
- Adam optimiser,
- batch size 1024 and 5 epochs.

The model scored articles in batches, and logits were passed through a sigmoid function to obtain click probabilities. MLPv1 provides a neural semantic ranker to compare against sparse TF-IDF features.

5.3.4. MULTI-LAYER PRECEPTRON WITH TF-IDF + DISTILBERT EMBEDDINGS (MLPV2)

The final and best-performing model was a hybrid neural ranker that combined compressed TF-IDF features with DistilBERT embeddings. This was tested because TF-IDF captures precise lexical overlaps and rare keywords, while BERT embeddings encode higher-level semantics. The model implementation followed:

1. **Dimensionality reduction for TF-IDF:** Truncated SVD with 64 components was applied on the sparse TF-IDF train matrix and projected on both training and validation articles.
2. **Feature concatenation:** For each article, its 64-dimensional TF-IDF SVD vector was concatenated with its 768-dimensional DistilBERT embedding, forming an 832-dimensional dense feature vector standardised with the learned scaler.
3. **Training data:** For two-stage ranking, all positive examples were kept, and hard negatives were sampled up to four times as many, capped at 400,000. This resulted in a training set of 21,685 pairs. Identified as a weakness in the previous model, this was implemented to mimic real-world data where positive examples are rare and more important.
4. **Model architecture:** input size 832, hidden layers 256 and 128 with ReLU activations, dropout of 0.2 after each hidden layer to reduce overfitting, and a final linear layer to a single logit.
Training used binary cross-entropy with logits, Adam optimiser learning rate, small weight decay, batch size 1024, and 5 epochs. Training loss steadily decreased from 0.38 to around 0.14, indicating successful convergence.

MLPV2 achieves a large improvement in two-stage Hit@5, MRR@5, and nDCG@5 compared to all other models, showcasing the benefit of combining TF-IDF and DistilBERT features with a more effective reranker training-sampling strategy.

5.4. Evaluation

The performance of the ranking models was assessed using three standard top-K recommendation metrics that quantified different aspects of ranking quality. All evaluation was computed at the impression level.

- **Hit@K** measures whether the true clicked article is retrieved anywhere in the top K items.

$$\text{Hit}@K = \frac{1}{N} \sum_{i=1}^N \mathbf{1}\{\text{clicked}_i \in \text{Top-}K_i\}$$

A higher Hit@K indicates stronger recall at small K values.

- **Mean Reciprocal Rank (MRR@K)** measures how early the clicked article appears in the top-K list.

$$\text{MRR}@K = \begin{cases} \frac{1}{\text{rank}} & \text{if the clicked article in Top-}K, \\ 0 & \text{otherwise.} \end{cases}$$

This penalises cases where the clicked article appears deeper in the list.

- **Normalised Discounted Cumulative Gain (nDCG@K)** applies a logarithmic discount to lower ranks. For binary relevance (clicked vs not clicked),

$$\text{DCG}@k = \sum_{j=1}^K \frac{y_j}{\log_2(j+1)}, \text{nDCG}@k = \frac{\text{DCG}@k}{\text{IDCG}@k}$$

It is maximal (1.0) only when the clicked article appears at rank 1, making it the most strict of the three metrics and a strong indicator of overall ranking precision.

6. Results

Candidate generation established the recall ceiling for all downstream rerankers (0.516).

Note: I tested two strategies for candidate generation. The earlier bucket-based one, i.e. select top categories → fill with popular news items within them → fill with similar articles to past clicks → stop at 400, for more diverse recommendations is not included in this paper. Substantial improvement (Recall@K 0.39 → 0.52) was achieved with the current sampler, confirming that global scoring provides more reliable coverage of the clicked article.

6.0.1. TWO-STAGE RANKING PERFORMANCE

Each reranker scored exactly the K=500 candidates per impression.

Traditional TF-IDF models (LR, XGBoost) achieved very

Model	Hit@5	MRR@5	nDCG@5
Logistic Regression (TF-IDF)	0.0586	0.0169	0.0268
XGBoost (TF-IDF)	0.0420	0.0163	0.0227
MLPV1 (BERT only)	0.0385	0.0177	0.0230
MLPV2 (SVD-TF-IDF + BERT)	0.6800	0.3993	0.4701

Table 1. Two-stage ranking performance at $K = 5$.

low Hit@5 and nDCG@5, indicating that they struggled

to meaningfully distinguish among hundreds of highly correlated, already-relevant items. MPv1 offered no improvement, showing that lexical features were critical for the task. MLPv2 outperformed all baselines and achieved 10x improvement, raising Hit@5 from 0.06 to 0.68. This strong performance is attributed to:

- The hybrid features combining lexical (TF-IDF SVD) and semantic (BERT) features, forming a more discriminative representation.
- The use of hard negatives generated from the candidate pool (e.g. topically similar, same category, same TF-IDF cluster) rather than random sampling.
- The above led to training that is fully aligned with the candidate distribution at evaluation.

6.0.2. ONE-STAGE RANKING PERFORMANCE

For completeness, ranking results over the impression pool (no candidate generation) were also evaluated.

Model	Hit@5	MRR@5	nDCG@5
Logistic Regression	0.4836	0.2301	0.2641
XGBoost	0.4707	0.2361	0.2642
MLP v1	0.4558	0.2363	0.2598
MLP v2	0.4864	0.2347	0.2662

Table 2. One-stage ranking performance at $K = 5$.

All models performed similarly, with Hit@5 around 0.46–0.49 and nDCG@5 around 0.26. The gap between models was much smaller because each impression contained only 20–50 items, many of which were irrelevant and linearly separable.

These results illustrate that classical models remained competitive with complex neural models under a one-stage setup due to their broad generalisation on sparse TF-IDF features. However, the two-stage architecture is more realistic for industry settings, and in that the hybrid MLP model delivers its largest gains.

7. Conclusions and Future Work

This project developed a complete, industry-inspired news recommendation pipeline using the MIND dataset, combining (i) a high-recall candidate generator and (ii) multiple rerankers trained on hybrid textual representations. I found that:

1. Candidate generation determines the ceiling.

The shift from bucket-based sampling to global scoring improved Recall@500 from 0.39 to 0.51, directly enabling stronger ranking performance.

2. Hybrid TF-IDF(BERT) features are essential.

TF-IDF captures high-dimensional lexical distinctions and BERT captures semantic similarity. Together, they form a robust representation space for ranking.

3. Training on hard negatives is decisive.

Rerankers trained on sampled random negatives underperformed. When trained with substantial hard negatives, mirroring industry practice, they achieved far higher discrimination and better performance.

4. Two-stage architecture is the right approach.

The system mirrors production recommenders such as YouTube, TikTok, and Microsoft News, where candidate recall and reranking models jointly determine performance.

5. Limitations.

- The model may underperform for cold-start users with little history or articles with very few clicks, and for extremely short or noisy articles that BERT and TF-IDF handle poorly.
- The setup is biased towards popular sources and categories. This may reduce exposure for niche categories, with less diverse recommendations hindering user exploration.
- The current approach uses only click history, and not richer information like time spent on an article or session structure. It also does not model temporal patterns with RNNs or transformers, which are increasingly common in industry and deliver better performance.

Even with these constraints, the two-stage recommendation approach demonstrates strong performance.

This work can be extended for improvements in both retrieval and ranking. Firstly, sequence-aware user modeling using architectures such as GRU4Rec, Transformers, or other temporal models could better capture evolving news preferences across sessions with different impressions. Secondly, graph-based representations (e.g., LightGCN or GraphSAGE) over the user–item interaction graph may also enhance candidate recall and solve cold-start issues. Finally, even though the bucket-based candidate generation approach delivered poor performance, it incorporated diversity and fairness objectives. Better methods to capture these objectives could counteract bias from a few sources or categories, leading to more balanced recommendation lists.

References

Li, L. and Li, T. SCENE: A scalable two-stage personalized news recommendation system. In *Proceed-*

ings of the 34th Annual ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 125–134. ACM, 2011. doi: 10.1145/2009916.2009936. URL <https://dl.acm.org/doi/10.1145/2009916.2009937>.

Qi, T., Wu, F., Wu, C., Huang, Y., and Xie, X. Pp-rec: News recommendation with personalized user interest and time-aware news popularity. *IEEE Access*, 9:80165–80174, 2021. doi: 10.1109/ACCESS.2021.3085226. URL <https://arxiv.org/abs/2106.01300>.

Wu, F., Qiao, Y., Chen, J., Wu, C., Qi, T., Lian, J., Liu, D., Xie, X., Gao, J., Wu, W., and Zhou, M. MIND: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 3597–3606, 2020. doi: 10.18653/v1/2020.acl-main.331. URL <https://aclanthology.org/2020.acl-main.331/>.

Wu, F., Qi, T., Wu, C., Huang, Y., and Xie, X. Fairrank: Fairness-aware single-tower ranking framework for news recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2387–2392, 2022. doi: 10.1145/3477495.3531842. URL <https://arxiv.org/abs/2204.00541#:~:text=FairRank%3A%20Fairness%20Aware%20Single%20Tower%20Ranking%20Framework%20for%20News%20Recommendation,-Chuhan%20Wu%2C%20Fangzhao&text=Single%20Tower%20models%20are%20widely,interest%20indicated%20by%20user%20behaviors>.