Solving cold start in news recommendations: a RippleNet-based system for large scale media outlet

Karol Radziszewski*
karol.radziszewski.dokt@pw.edu.pl
Warsaw University of Technology
Warsaw, Poland
Ringier Axel Springer Tech
Warsaw, Poland

Michał Szpunar
michal.szpunar@ringieraxelspringer.pl
Ringier Axel Springer Tech
Warsaw, Poland

Piotr Ociepka
piotr.ociepka@ringieraxelspringer.pl
Ringier Axel Springer Tech
Krakow, Poland
AGH University
Krakow, Poland

Mateusz Buczyński
mp.buczynski2@uw.edu.pl
Ringier Axel Springer Tech
Warsaw, Poland
University of Warsaw
Warsaw, Poland

Abstract

We present a scalable recommender system implementation based on RippleNet, tailored for the media domain with a production deployment in Onet.pl, one of Poland's largest online media platforms. Our solution addresses the cold-start problem for newly published content by integrating content-based item embeddings into the knowledge propagation mechanism of RippleNet, enabling effective scoring of previously unseen items. The system architecture leverages Amazon SageMaker for distributed training and inference, and Apache Airflow for orchestrating data pipelines and model retraining workflows. To ensure high-quality training data, we constructed a comprehensive golden dataset consisting of user and item features and a separate interaction table, all enabling flexible extensions and integration of new signals.

CCS Concepts

• **Information systems** → **Personalization**; **Recommender systems**; *Language models*; • **Computing methodologies** → **Knowledge representation and reasoning**.

Keywords

Recommender Systems, Cold-start Problem, RippleNet, Knowledge Graphs, Knowledge-aware Recommendation, News Recommendation

1 Introduction

The rapid pace of content publication in news and media platforms introduces significant challenges for recommender systems, particularly due to the continuous emergence of new items and sparse user interactions. These challenges, commonly known as the cold-start problem, limit the effectiveness of traditional approaches that rely heavily on historical interaction data. To bridge this gap, we propose an enhanced recommendation approach leveraging RippleNet, a knowledge-aware recommender framework. Our solution integrates RippleNet's entity embeddings with text embeddings generated by large language models (LLMs), enabling effective semantic representation and recommendation of newly introduced content, hence addressing the cold-start limitations visible in highly dynamic domains such as news recommendation.

2 Literature review

2.1 Cold start Problem

A fundamental challenge in recommendation is the cold start problem, which occurs when no observable data about the user or an item is available in the past [2]. On the user's end, a practical fallback is to recommend generally popular or trending items until more personalized signals are collected. In new item cold-start cases, a freshly added item has no interactions yet, most commonly content-based techniques are used, since item attributes or content can serve as latent space, which "already solves a major part" of the problem [3]. Metadata-driven approaches (using item descriptors like genre, tags, or knowledge from knowledge graphs) and hybrid models (which combine content-based scores with collaborative signals) are widely used to give new items an initial score.

Another angle is to use the exploration-exploitation tradeoff: for instance, deploying a bandit algorithm that occasionally promotes new items to gather feedback. However, when only implicit feedback (clicks, impressions, etc.) is available, there are no explicit negative signals, making it harder to discern a new user's dislikes. In highly dynamic domains like news and media streaming, new items arrive continuously and have short lifespans. This creates a permanent item cold-start scenario [9]. Conventional collaborative filtering models (which assume a relatively static item catalog) struggle here [5]. Research has addressed this by frequent model updates and by prioritizing recent information.

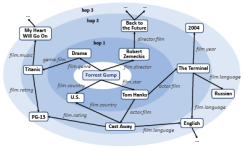
*Corresponding author

2.2 Knowledge-Aware Recommendations

In recent years, knowledge-aware recommender systems have gained attention as a solution to data sparsity and cold-start issues. The key idea is to incorporate side information in the form of knowledge graphs (KG) about items and users. Traditional approaches fell into two categories: (1) embedding-based methods, which learn latent representations of entities (items and their related attributes); (2) path-based methods, which explicitly search for connectivity paths between users and items via intermediate knowledge nodes. While effective to a degree, each has limitations: embedding-based models may not capture complex relational semantics, and path-based models can be ad-hoc or hard to scale.

RippleNet [10] emerged as an influential knowledge-aware framework designed to overcome these limitations. It treats the knowledge graph as a base of interconnected entities and relations (e.g. items, their attributes, categories, actors, etc.). The model introduces a "ripple" propagation mechanism inspired by the ripple effect on water. Given a user's known interests, RippleNet activates a set of entities in the knowledge graph that are directly linked to those interests (first-hop neighbors). These might be, for example, the authors, topics, or brands associated with the user's consumed items. Then, in an iterative fashion, the model expands further (second hop, third hop, etc.), activating entities connected to the first set, and so on (this process is graphically visualized in the figure 1). Each "ripple" thus spreads the user's preference signal further through the graph. By the end of H hops, multiple waves of activated entities – triggered by different historical items – collectively form a rich preference distribution over the knowledge graph. For a given candidate item to recommend, RippleNet estimates the user's preference by how strongly these ripples intersect with that item's knowledge profile. This end-to-end architecture showed substantial accuracy gains across several domains, including movies, books and even news recommendation, outperforming earlier baselines.

Figure 1: RippleNet's ripple sets in a knowledge graph from exemplary dataset



Source: *Wang et al., RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems, 2018*

Knowledge-aware models are especially beneficial in cold-start situations; for example, even if a new item has no users yet, its connections in a knowledge graph (such as its category, creator, or related items) can be used to find potential interested users. Likewise, a new user's few initial clicks can be mapped to knowledge entities (topics, etc.) to quickly broaden the understanding of their interests. They have also been applied in news recommendation. For example, [11] proposed DKN: Deep Knowledge-Aware Network. It combines news text understanding with entity information from a knowledge graph. It employs a convolutional neural network where each news article is represented by two aligned sequences: words (in the title/body) and entity mentions (mapped to a knowledge graph embedding) - the model learns a fused semantic-knowledge representation of news content. An attention module then aggregates a user's reading history in a personalized way, emphasizing those past articles most relevant to the candidate news.

2.3 Recommender Systems for the News Domain

Recommender systems in news and media face distinct challenges due to the highly dynamic nature of the content, characterized by continuous publication and rapid obsolescence of news articles. Unlike static domains such as movies or books, this domain perpetually operates in a state of item cold-start, demanding strategies that emphasize recent and trending content. Classic collaborative filtering approaches struggle under these conditions due to sparse interaction data and the necessity for rapid model updates. To manage this, leading platforms such as Google News [1] developed scalable, hybrid approaches combining clustering techniques, latent semantic models, and real-time co-visitation metrics to continuously adapt recommendations to user interactions and item freshness.

Addressing these challenges, our approach employs RippleNet to capture deeper semantic relationships among items. Specifically, we enhance RippleNet by mapping embeddings learned during model training directly onto text embeddings generated by large language models (LLMs). This allows for effective representation of new items, significantly mitigating cold-start issues by capturing inherent semantic content similarities even without user interaction history.

3 Data

We constructed "golden" RASP recommendation dataset in the unified Recbole [12] unified format, comprising three core files:

- `.inter`, which logs user–item interactions and interaction-related features;
- `.user`, detailing user attributes;
- `.item`, encompassing item metadata.

To support scalability research, three dataset versions - small, medium, and large - are provided, with their descriptive statistics summarized in Table 1. Notably, this dataset is constructed from real-world interactions in the Polish news service and Polish language, a Slavic language family member, unlike many existing datasets based on English. This linguistic distinction introduces unique semantic, morphological, and lexical challenges relevant to natural language processing and recommendation modeling, particularly in content-based and embedding-based approaches. We added Stylometrix-based features [6] to catch this linguistic domain. The dataset spans from March 12, 2025, to April 14, 2025, and is split into training, validation (the three days preceding the test period), and testing (the final three days) subsets.

The dataset's `.inter` file contains interaction-level data, including: (i) `user_id` and `item_id` identifiers, (ii) `is_click`, the binary recommendation target (1 for click, 0 for impression only), (iii) temporal features such as `event_timestamp_unix`, `event_date`, `weekday`, `hour`, and `is_business_day` (based on Polish holidays), (iv) engagement metrics like `ip_cnt`, `pu_ip_cnt`, `pv_cnt`, `glowna_ip_cnt`, `pv_on_content_publication_premium_cnt`, and `ses_`

duration_sum, (v) device and session features such as device_type, context_client_brand, context_client_version, cs (screen size), and rh_user_agent, (vi) geolocation approximations via latitude, longitude, accuracy_radius, and regional identifiers (geoip_city_name, geoip_region_name), (vii) user state flags, e.g., is_active_click, is_active_pageview, user_evoked_sso_logged_in, and user_ subscriber, (viii) a predictive feature like lts_pred (likelihood to subscribe to Onet Premium).

The .user file user profile information, such as: (i) user BERT-based reading preferences for 10 categories calculated as precentage of read articles within a category, (ii) a login status (user_sso_name), (iii) user browser and device.

The .item file captures rich article-level content and metadata, including: (i) text-based fields: title, lead, text, and derived metrics like text_length, (ii) authorship and multimedia: author, images, (iii) semantic annotations via wikidata_entities_words, wikidata_ entities_ids, wikidata_entities_scores, wikidata_topics, and wikidata_topics_scores, (iv) premium content status: content_publication_premium, (v) neural embeddings (openai_embedding, 256-dimensional, computed via OpenAI's *text-embedding-3-large*), (vi) internal BERT-based category confidence scores for 10 categories, (vii) stylometrix features for title, lead, and text, capturing linguistic, syntactic, and lexical complexity of Polish language.

The "golden" label reflects the dataset's high quality and rigorous preprocessing. By aligning with Recbole's format, the dataset enables immediate compatibility with numerous state-of-the-art recommendation models and tools, fostering reproducibility and streamlined benchmarking within RASP's recommendation system.

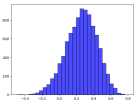
Table 1: Dataset statistics for RASP Recommendation Data, including interaction labels. Last two columns shows number of positive (clicks) and negative (recommendations without click) interactions.

Dataset	Split	Users	Items	Pos (1)	Neg (0)
Small	Train	18,727	20,794	498,973	4,823,570
	Valid	8,033	3,653	62,496	655,422
	Test	7,560	2,845	52,160	456,302
Medium	Train	74,743	20,800	1,975,109	18,814,474
	Valid	32,393	3,653	247,652	2,616,881
	Test	29,941	2,846	208,294	1,815,687
Big	Train	298,526	20,800	7,900,163	75,702,502
	Valid	129,146	3,653	988,923	10,411,131
	Test	119,654	2,846	827,126	7,233,277

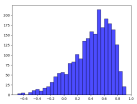
4 Cold-start solution

The biggest challenge in addressing the cold start problem in the RippleNet model used for news recommendations is its inability to recommend entities (i. e., articles) which were unknown or non-existent during the model training. To overcome this limitation, we developed a method for translating semantic information about a new article into the representation space of the RippleNet model. Since the RippleNet model represents each entity as an internal embedding and our knowledge about semantics of each article is based on embeddings generated by Large Language Models (LLMs) — such as OpenAI [7] models or domain-specific alternatives like PolBERT [4] — we decided to verify two distinct approaches, one based on neural encoder and second based on replacing unknown embeddings with the closest known ones.

Figure 2: Histogram of cosine similarity between matched and real embedding



(a) using neural encoder



(b) using cosine similarity matching

4.1 Neural encoder

The first approach was to train a neural encoder to predict the RippleNet embedding from the corresponding LLM-based embedding. The encoder was trained as a fully connected feedforward network with two hidden layers. We evaluated two loss functions, mean squared error (MSE) and cosine similarity. As the latter yielded better results in terms of alignment with the RippleNet space, we decided to ditch MSE. The training and evaluation datasets consisted of pairs of both RippleNet and corresponding LLM embeddings of articles present in the RippleNet training set. Exemplary results are in figure 2. It's a histogram of cosine similarities between trained embeddings and encoded LLM ones.

4.2 Similarity-based replacement

In contrast to the neural encoder approach, the similarity-based replacement method avoids the use of artificial non-existent embeddings as input to the RippleNet model. Instead, it computes the cosine similarity between the LLM-based embedding of the new item and the LLM-based embeddings of all known items. The RippleNet embedding corresponding to the most similar known item is used as input to the recommender model. Exemplary results are in figure 2. We notice a larger number of instances close to real embedding, hence we chose this setting for further testing.

5 Methodology

5.1 Offline evaluation

When evaluating offline the similarity-based recommendation solution - which is better suited for deployment in large-scale online environments - we use our production model as a baseline. The production system employs a deep neural network for click prediction and is described in our prior work [8]. Since RippleNet requires retraining daily to remain effective in dynamic domains such as news recommendation, we performed an offline evaluation by training it on a single day and testing it across three temporal slices: the day before training, the training day itself and the day after.

5.2 Online evaluation

The news recommendation environment evolves rapidly, necessitating daily retraining of the RippleNet model to continuously integrate new user-item interactions and enable its evaluation in an online setting. We orchestrate the retraining process using an

Airflow DAG, which automates both data processing and model training tasks executed on AWS SageMaker. The overall workflow is depicted in Figure 3. The pipeline begins with a data availability check, which ensures that data is present. Once the prerequisite data is available, next task extracts knowledge graph and reformat data to match Recbole format. The preprocessed data is then consumed by two parallel tasks: training, which performs model training on GPU using the sample of latest data, and user profiles creation, which constructs ripple sets at the scale of millions of users. Their outputs are aggregated in the merge task. Next, the torch-model-archiver module packages the trained model into a format suitable for deployment. Finally, the deployment task, deploys the archived model to the production environment, enabling real-time inference and online evaluation.

Figure 3: The pipeline illustrates the automated training and deployment process for the RippleNet model.

Orange nodes represent Airflow tasks, yellow nodes correspond to SageMaker processing jobs, and the green node denotes the SageMaker training job.

6 Results

6.1 Offline evaluation

The best-performing neural encoder variant, when integrated into the RippleNet model, achieved NDCG@10 of 0.0004, Precision@10 of 0.0002, and Recall@10 of 0.0004. These results indicate that the encoder-based approach fails to meet the effectiveness requirements of our recommendation scenario. Consequently, we proceeded to evaluate a similarity-based solution.

The RippleNet model was configured with the following RippleNet hyperparameters: `n_hop` = 5, `n_memory` = 16, learning rate = 0.01, and trained for up to 50 epochs with early stopping based on NDCG@10 (patience = 5). These parameters were selected via a hyperparameter tuning process with Recbole library, optimizing NDCG@10 metric, without similarity-based solution.

Table 2 presents NDCG@10, Precision@10, and Recall@10 for both models across the evaluated days. The production baseline outperforms RippleNet across all metrics and dates. RippleNet shows highly variable performance: it performs best on the training day (e.g., NDCG@10 of 0.02602), but its effectiveness deteriorates rapidly before and after training. Especially after training it is 3-4 times worse than on the day before training (e.g., NDCG@10 0.00302 vs 0.00914)

This bad peformance indicates that RippleNet is only effective on the day it is trained. One possible explanation is that the model heavily relies on direct knowledge propagation through entities seen during training. On the training day, most of the test-time articles are already present in the model's graph structure, minimizing reliance on the similarity-based fallback mechanism. However, when tested on unseen content from adjacent days, the model struggles to generalize - exposing its dependence on memorized entities and possible limited use of similarity-based reasoning during inference.

Table 2: Performance of production baseline and RippleNet with similarity-based solution across three evaluation days and three ranking metrics.

Model	NDCG@10			Precision@10			Recall@10		
	Train - 1	Train	Train + 1	Train - 1	Train	Train + 1	Train - 1	Train	Train + 1
Production baseline	0.01909	0.02790	0.02199	0.01127	0.01420	0.01068	0.03551	0.04313	0.03118
RippleNet + Similarity	0.00914	0.02602	0.00302	0.00606	0.01443	0.00137	0.01683	0.04080	0.00455

6.2 Online evaluation

Based on our previous experience, offline evaluation metrics do not necessarily correlate with online performance. Therefore, we conducted an online evaluation of the model and its deployment pipeline. The test was carried out over a two-day period, during which we trained two separate similarity-based models: one tailored for mobile users and the other for desktop users, reflecting distinct engagement patterns across platforms.

During the online test, the observed results were consistent with offline metrics. We recorded engagement uplifts of -13.77% on mobile and -16.21% on desktop of our business metric, indicating alignment between offline and online evaluation signals.

7 Conclusions

In conclusion, our proposed method effectively addresses the cold-start challenge in news recommendation by integrating RippleNet with semantic embeddings derived from large language models. This hybrid approach significantly enhances the ability to recommend newly published content without prior interaction data, ensuring timely and relevant content delivery. The empirical findings suggest that leveraging semantic relationships captured by RippleNet and LLM embeddings provides a foundation for personalized recommendations in rapidly changing environments, however at the current state such setup does not reach production-grade performance. Future research could explore further refinements to embedding integration techniques.

References

[1] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web* (Banff, Alberta, Canada) *(WWW '07)*. Association for Computing Machinery, New York, NY, USA, 271–280. doi:10.1145/1242572.1242610

[2] Toon De Pessemier, Bruno Willems, and Luc Martens. 2025. Active learning algorithm for alleviating the user cold start problem of recommender systems. *Scientific Reports* 15, 24493 (July 2025). doi:10.1038/s41598-025-09708-2

[3] Mozhgan Karimi, Dietmar Jannach, and Michael Jugovac. 2018. News recommender systems – Survey and roads ahead. *Information Processing & Management* 54, 6 (2018), 1203–1227. doi:10.1016/j.ipm.2018.04.008

[4] Dariusz Kłeczek. 2020. PolBERT: Attacking Polish NLP Tasks with Transformers. In *Proceedings of the PolEval 2020 workshop*. 79–88.

[5] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web (WWW '10)*. ACM, 661–670. doi:10.1145/1772690.1772758

[6] Inez Okulska, Daria Stetsenko, Anna Kołos, Agnieszka Karlińska, Kinga Głąbińska, and Adam Nowakowski. 2023. StyloMetrix: An Open-Source Multilingual Tool for Representing Stylometric Vectors. arXiv:2309.12810 [cs.CL] https://arxiv.org/abs/2309.12810

[7] OpenAI. 2024. *New embedding models and API updates*. https://openai.com/index/new-embedding-models-and-api-updates/ , last visited on 2025-07-17.

[8] Karol Radziszewski and Piotr Ociepka. 2024. Enhancing Prediction Models with Reinforcement Learning. In *Proceedings of the 12th International Workshop on News Recommendation and Analytics (INRA'24) (CEUR Workshop Proceedings, Vol. 3929)*, Vandana Yadav (Ed.). CEUR-WS.org, Bari, Italy, –. https://ceur-ws.org/Vol-3929/short4.pdf

[9] Martin Saveski and Amin Mantrach. 2014. Item cold-start recommendations: learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender systems*. 89–96.

[10] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (Torino, Italy) *(CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 417–426. doi:10.1145/3269206.3271739

[11] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep Knowledge-Aware Network for News Recommendation. arXiv:1801.08284 [stat.ML] https://arxiv.org/abs/1801.08284

[12] Lanling Xu, Zhen Tian, Gaowei Zhang, Junjie Zhang, Lei Wang, Bowen Zheng, Yifan Li, Jiakai Tang, Zeyu Zhang, Yupeng Hou, Xingyu Pan, Wayne Xin Zhao, Xu Chen, and Ji-Rong Wen. 2023. Towards a More User-Friendly and Easy-to-Use Benchmark Library for Recommender Systems. In *SIGIR*. ACM, 2837–2847.