Recommending Interesting Writing

Rohan Bansal
The Browser
rohan@thebrowser.com

Uri Bram
The Browser
uri@thebrowser.com

Robert Cottrell
The Browser
robert@thebrowser.com

Jaan Altosaar
Princeton University
altosaar@princeton.edu

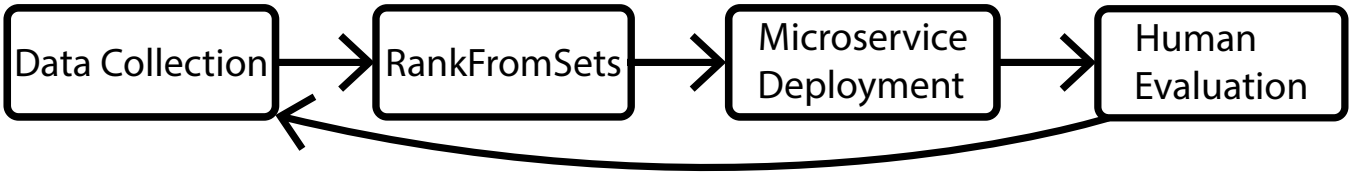

Figure 1: End-to-end pipeline for recommending nonfiction writing to editors at The Browser. Positive examples for the RANKFROMSETS recommendation model [2] are collected from editors' history of curated articles, and negative examples from news sources. After training and offline evaluation of the recommendation model, it is deployed as a microservice, and editors' feedback on the recommendation performance is used to inform refinement of data collection, training, and architectural choices in the recommendation model.

ABSTRACT

We describe a system for recommending nonfiction writing to editors at The Browser, a curation service for interesting writing. The editors' goal is select articles from a large list of candidates, and these selections of nonfiction writing are shared with subscribers. To aid the editors, we build a recommender system that classifies articles based on their content. The recommendation model is RANKFROMSETS (RFS), chosen for its scalability and explainability, with architectures that allow editors to understand which words in an article informed a recommendation [2]. Further, editors can choose which latent features of articles to upweight. We train RFS on historical data, and show that this translates to good performance in an online setting with qualitative feedback from editors on unseen candidate articles. Due to resource constraints, we deploy RFS using a microservices architecture on a cloud computing platform. For reproducibility and transparency of the user-facing system, we open source the end-to-end pipeline and release a demo[1], data collection, training and deployment scripts, and model parameters.[2]

CCS CONCEPTS

• **Applied computing** → **Document searching**; • **Computing methodologies** → **Learning from implicit feedback**.

[1]https://the-browser.github.io/recommending-interesting-writing/
[2]https://github.com/the-browser/recommending-interesting-writing

KDD 2020, August 23–27, 2020,

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

KEYWORDS

content-based recommendation, open source, user interface

ACM Reference Format:

Rohan Bansal, Uri Bram, Robert Cottrell, and Jaan Altosaar. 2018. Recommending Interesting Writing. In *KDD '20: International Workshop on Industrial Recommendation Systems, August 23–27, 2020*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Creative nonfiction, longform journalism, and blog posts are examples of the types of articles curated by The Browser's team of editors. The editors filter through a large number of articles from various publications to select content to recommend to subscribers.

Journalism and content curation services such as The Browser can be constrained by financial resources, and this affects the types of machine learning solutions that might be feasible to aid editors in filtering through large numbers of articles. For example, serving models on GPUs may be too expensive, and a trade-off must be made between cost and performance.

Similar to the trade-off in performance due to constraints, there is a trade-off in user privacy intrinsic to recommender systems. A machine learning model must be trained on historical consumption data, and for a performant model, the items in the training data should be similar to new content the recommender is designed to filter. However, the incentive structures of operating a recommender system within a business can influence decisions around privacy and transparency [5]. For example, advertising business models may engender recommender systems that upweight attention-grabbing content and hence time spent looking at ads. Such content might maximize a user's time spent with the service over time at the expense of long-term user experience or consent. Outside of recommender systems, privacy-preserving and open source tools

such as the Signal encrypted messaging service[3] may provide improved user experience in terms of privacy-preserving, transparent, and explainable algorithms [3].

We build an end-to-end recommender system to address two aims: (1) to recommend interesting writing to editors at The Browser that provides interpretable recommendations and (2) to release a lightweight, open-source machine learning framework for developing and deploying recommender systems for quality writing.

2 METHOD

We describe the recommendation model and the pipeline for data collection, training of the model, and evaluation. This is illustrated in Figure 1.

Recommendation Model. RANKFROMSETS (RFS) is a recommendation model suitable for our problem. It is scalable to large numbers of articles and words, and can maximize the evaluation metric of recall [1, 2]. Recall, or the fraction of true positives returned by a recommendation model, is an appropriate evaluation metric for recommending interesting writing to editors at The Browser. A recommendation model such as RFS can be readily backtested with recall as an evaluation metric, as historical data contains positive examples (articles selected by the editors) but rarely contains negative examples (articles seen but not selected by the editors).

RFS is a recommendation model defined by a binary classifier. For a user u and item m with attributes x_m (the set of unique words in an article), RFS is described by the probability of $y_{um} = 1$ (user u consuming item m):

$$p(y_{um} = 1 \mid u, m) = \sigma\left(f\left(u, x_m\right)\right) ,$$

where σ is the sigmoid function. To parameterize the binary classifier in RFS, we use an inner product architecture:

$$f\left(u, x_m\right) = \theta_u^\top \left(\frac{1}{|x_m|} \sum_{j \in x_m} \beta_j \right) . \tag{1}$$

In this architecture, the user embedding θ_u includes a dimension that is fixed to unity. Word embeddings β_j (including a bias dimension for every word) and the publication embedding are fit with maximum likelihood estimation, and negative examples are sampled uniformly at random to balance positive examples.

Data Collection and Preprocessing. For positive examples, we use the historical set of articles curated by editors at The Browser. We augment the training data with articles selected by the editors of other curation services, and treat all positively-labeled examples curated by editors as data from a single user due to a paucity of data. We use articles from news websites as examples with negative labels, and collect additional articles with negative labels from websites most-featured by the editors to mimic the editorial process of reading a large swath of articles in a feed and distilling an article list to a select few. For preprocessing the data we use the tokenizer released by Devlin et al. [4] and discard words not recognized by the tokenizer. This procedure leaves a dictionary with 30k words, and 148k training examples with 28k positive labels.

[3]https://signal.org/

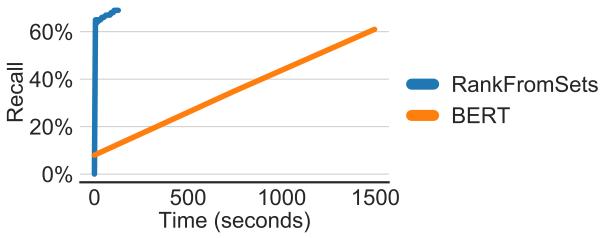

Figure 2: RANKFROMSETS outperforms models based on word embeddings permutation-marginalized recurrent neural networks (denoted by LSTM) on meal recommendation. The recommendation models are trained on data from a food tracking app as described in ?? and are evaluated using the sampled recall metric, ??. The inner product, neural network, and residual regression functions for RANKFROMSETS are in ??????.

Empirical Study. Performance was assessed with recall, and 15% of the data was held-out for validation and test sets. The performance of the model was similar for large embedding sizes, and we selected the dimension of the embeddings to be 25. We cross-validate using the RMSProp optimizer [6] and grid search over learning rates of $\{10^{-1}, 10^{-3}, 10^{-4}, 10^{-5}\}$ and select the best-performing model for deployment.

3 EVALUATION

We study the performance of RANKFROMSETS on a dataset that we collect to assess whether RFS, in addition to providing interpretable recommendations to help editors make decisions about articles.

Data collection.

Experimental setup: RankFromSets.

Experimental setup: BERT. for rankfromsets we searched over 6 embedding sizes [10,25,50, 100, 500, 1000] and 5 batch sizes [500, 1000, 2000, 5000, 10000]. We tried two optimizers, RMS and SGD, both with a momentum of 0.9 and learning rates of [1e-2, 1e-3, 1e-4, 1e-5] for RMS, while learning rates for SGD varied based on batch size and average token count Breakdown of Data: Train - Total: 100797 Positive: 18598 Negative: 82199 Test - Total: 272448 Positive: 4049 Negative: 268399 Evaluation - Total: 272447 Positive: 4039 Negative: 268348 Both models used the same randomly selected 50k articles from the evaluation set for validation, and the entire test set was used at the end to generate predictions on new data.

Evaluation. Qualitatively, editors at The Browser preferred the recommendations of our pipeline over their current workflow of a reading list sorted in terms of recency and use the system in production.

4 DEPLOYMENT AND USER INTERFACE

As curation services can be constrained by computational and financial resources, we choose to deploy the RFS model as a cloud computing-based microservice. Further, we exploit the properties

Model	Recall @ 10 (%)	Recall @ 100 (%)
RFS	0.32	2.54
BERT (BERT)	**0.44**	2.54

Table 1: with the entropy proximity statistic improves top-10 out-matrix recall of RFS fit to arXiv user behavior data. We report the recall for items with no clicks in the training data (described in ??) for the best-performing settings of both and RFS. Recall at 100 recommendations is comparable between the methods.

of the the RFS architecture in Equation (1) to aid explainability and exploration. For explainability, we display words j with high (low) inner product $\theta_u^\top \beta_j$, as these words contribute the most (least) to a prediction of a positive label. To enable users to explore patterns in a large list of articles, we enable them to modulate the dimensions of θ_u with the greatest weight. After a user has scaled these dimensions according to their preference, the recommendation results are returned by the microservice using Equation (1).

5 DISCUSSION

Future Work. Comparing to transformer-based models [4] would help clarify how much performance is given up when choosing an efficient model such as RFS for recommendations. For explainability beyond top words in an article contributing to a recommendation, it would be interesting to develop methods for enabling editors to search through interpretable latent dimensions. Building quantitative human-in-the-loop evaluation methods into a user interface may also help guide modeling choices. Finally, studying whether the transparency provided by open-source recommendation systems can improve user experience is an open problem.

ACKNOWLEDGMENTS

The authors are grateful to Christian Bjartli for help with data collection.

REFERENCES

[1] Jaan Altosaar. 2020. *Probabilistic Modeling of Structure in Science: Statistical Physics to Recommender Systems.* Ph.D. Dissertation. Princeton University.
[2] Jaan Altosaar, Wesley Tansey, and Rajesh Ranganath. 2020. RankFromSets: Scalable Set Recommendation with Optimal Recall. *American Statistical Association Symposium on Data Science & Statistics* (2020).
[3] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. 2017. A Formal Security Analysis of the Signal Messaging Protocol. In *2017 IEEE European Symposium on Security and Privacy (EuroS P).* 451–466.
[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Association for Computational Linguistics.* Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423
[5] Nicholas Diakopoulos. 2020. *Oxford Handbook of Ethics and AI.* Oxford University Press, Chapter Accountability, Transparency, and Algorithms.
[6] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning* (2012).