Recommending Interesting Writing

Rohan Bansal
The Browser
rohan@thebrowser.com

Jordan Olmstead
The Browser
jordan@thebrowser.com

Uri Bram
The Browser
uri@thebrowser.com

Robert Cottrell
The Browser
robert@thebrowser.com

Jaan Altosaar
Princeton University
altosaar@princeton.edu

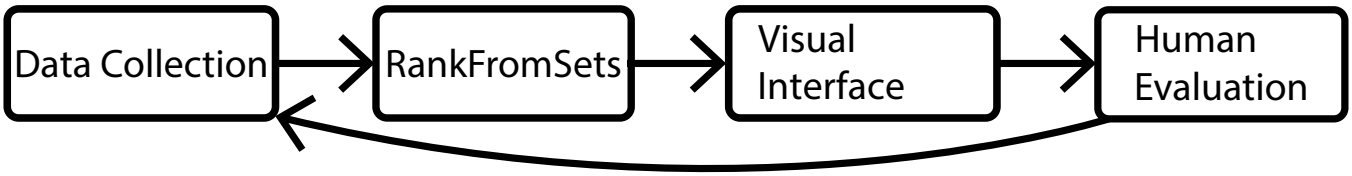

Figure 1: End-to-end pipeline for recommending writing to editors at The Browser. The RANKFROMSETS recommendation model [2] is trained on data consisting of positive examples and negative examples collected from the editors' history of curated articles, and negative examples from news sources. After training and offline evaluation of the recommendation model, RANKFROMSETS is deployed as a microservice on AWS Lambda, with the visual interface hosted on Github Pages. Editors can control the recommender system using the visual interface, which can aid in their decision-making. The editors' interrogation of the recommendation model and feedback is then used to inform of data collection and training.

ABSTRACT

We build a visual interface for recommending articles to editors at The Browser, a curation service for interesting writing. From a large list of candidates, editors decide which articles are selected and shared with subscribers. To aid the editors in this decision-making task, we build a visual interface for a recommendation model, RANKFROMSETS (RFS) [2], that classifies articles based on their words. Control of the recommendation model is built into the visual interface. For example, an editor can use a topic slider to receive a new list of recommendations according to topical words in articles. These topic sliders might be used to increase or decrease the ranking of articles with words related to crime, business, or technology. The visual interface is also designed to be explanation-aware: words that contribute positively or negatively to an article's ranking are displayed. For the backend of the visual interface, RFS is trained on historical data. In an offline empirical study, we find that RFS outperforms BERT [4], a competitive classification model, in terms of recall. Further, we measure RFS to be 10 times faster to train and to return predictions 2000 times faster than BERT. These are beneficial properties for the visual interface, and we demonstrate that RFS can be deployed on the free tier of AWS Lambda using a short python script and numpy dependency. For reproducibility, transparency, and trust of the visual interface, we open source and release a public demonstration,[1] data collection, training and deployment scripts, and model parameters.[2]

CCS CONCEPTS

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

KEYWORDS

content-based recommendation, open source, visual interface

ACM Reference Format:

Rohan Bansal, Jordan Olmstead, Uri Bram, Robert Cottrell, and Jaan Altosaar. 2020. Recommending Interesting Writing. In *RecSys '20: Workshop on Interfaces and Human Decision Making for Recommender Systems, September 22–26, 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 read a large number of articles from various publications to select content to recommend to subscribers.

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

RecSys 2020, September 22–26, 2020,

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

In building a recommender system to help editors sift through many documents, it is motivating to highlight the trade-off in user privacy intrinsic to recommender systems. A machine learning model must exploit information about a user. However, the incentive structures of operating a recommender system within a business can influence decisions around privacy and transparency [5]. For example, business models that rely on online advertising 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 a 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 and visual interfaces [3]. Due to poor incentive structures for releasing recommender systems and visual interfaces that exploit private information about users, there are few examples of end-to-end pipelines for recommending content to users. This serves as motivation to build and deploy a recommendation model and corresponding explanation-aware visual interface to give users control, and inform them about how data is being used to make recommendations.

We build an end-to-end recommender system visual interface to address two aims: (1) to aid editors at The Browser in their decision-making task, and give them control through an explanation-aware interface, and (2) to release a lightweight, performant, open-source visual interface framework for developing and deploying explanation-aware recommender systems for document recommendation. In an offline evaluation, we show that the recommendation model we use for the visual interface outperforms BERT, a competitive document classification model. In a qualitative study, the control and explanations provided by the visual interface help editors in their decision-making and can help find bugs in the recommendation model.

2 RECOMMENDATION MODEL

RANKFROMSETS (RFS) is the recommendation model that powers the visual interface; the main part of the pipeline illustrated in Figure 1. RFS scales to large numbers of articles, 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). Further, as our goal is to build an explanation-aware visual interface that can also serve to control recommendations, RFS is fast, interpretable, and simple to integrate into a user interface.

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.

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 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.

Metrics. Performance of the recommendation models is assessed with recall, and 15% of the data is held-out for validation and test sets.

Experimental setup: RankFromSets. 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.

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.

for BERT, used AdamW optimizer with a linear scheduler, with 4 different learning rates [2e-5, 3e-5, 4e-5, 5e-5] and a batch size of 32. Tried four different amounts for warmup steps [100, 1000,

[3] https://signal.org/

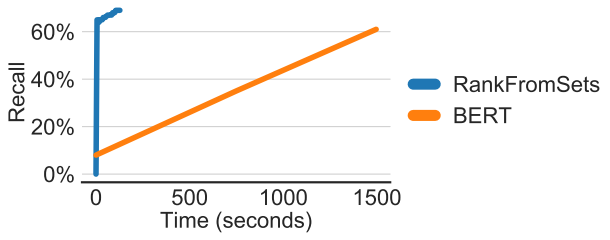

Figure 2: RANKFROMSETS outperforms models based on word embeddings permutation-marginalized recurrent neural networks (denoted by LSTM) on meal recommendation. The recommendation models are trained on data from a food tracking app as described in ?? and are evaluated using the sampled recall metric,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

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.

1000, 10000] and total training steps were within the set [500, 5000, 50000, 100000] depending on the warmup step amount First 510 tokens were taken from the article to generate predictions so as to fit the BERT length requirements

BERT 466/1000 rankfromsets 531/1000

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 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).