Recommending Interesting Writing using a Controllable, Explanation-Aware Visual Interface

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

Gabriel Reder
Stanford University
gkreder@stanford.edu

Jaan Altosaar
Princeton University
altosaar@princeton.edu

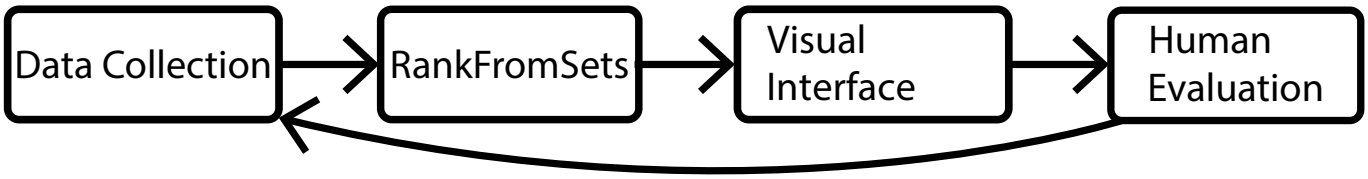

Figure 1: End-to-end pipeline for recommending writing to editors at The Browser with a controllable, explanation-aware visual interface. The RANKFROMSETS recommendation model [2] is trained on data consisting of positive examples from the editors' history of curated articles, and negative examples from news sources. After training and offline evaluation of the recommendation model, RANKFROMSETS is deployed as a microservice on Amazon Web Services 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 informs further 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. This speed is a beneficial property 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]

Author Keywords

content-based recommendation, open source, visual interface

CCS Concepts

•**Applied computing** → **Document searching;** •**Computing methodologies** → **Learning from implicit feedback;** Please use the 2012 Classifiers and see this link to embed them in the text: https://dl.acm.org/ccs/ccs_flat.cfm

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.

In building a recommender system to help editors sift through many documents, it is motivating to highlight the trade-off in

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

IntRS '20 - Joint Workshop on Interfaces and Human Decision Making for Recommender Systems, September 26, 2020, Virtual Event

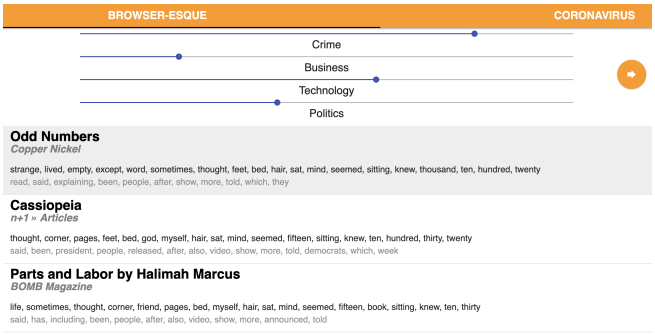

Figure 2: Visual interface to RANKFROMSETS includes topic sliders for setting user preferences as well as most important topic words for each found article

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. In comparison, 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]. But the incentive structures for releasing recommender systems and visual interfaces that exploit private information about users are poor. There are few examples of end-to-end, open source, free-to-deploy pipelines for recommending content to users using a visual interface. This motivates building and deploying 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 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 help find bugs in the recommendation model.

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, and RFS is fast, interpretable, and simple to integrate into a user interface as we describe later.

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 [1].

VISUAL INTERFACE

The visual interface is designed with RFS as the backend recommendation model. We describe how the inner product architecture for RFS enables a visual interface that is interpretable to provide explanations for why an item is recommended, and enables control so users can filter recommendations to help with decision-making.

Explanation-aware recommendation

The user embedding θ_u and word embeddings β_j in Equation (1) can be used to interpret a recommendation. The logit for a given document with a set of words x_m is the sum of per-word logits, which are computed as the inner product of the user embedding and word embedding. The per-word contribution of a word in a document to the logit that determines the document's ranking in a list of recommendations is

$$w_{uj} = \theta_u^\top \beta_j . \tag{2}$$

This weight w_{uj} helps explain why a document was recommended, using information about both the user u and the word j. In the visual interface, words in a document are first sorted by their contributions to a document's logit w_{uj}, and the top words are displayed. Similarly, words that lower a document's ranking are also displayed, to inform a user of which words detract from the recommendation of a document.

Interface for controlling recommendations

In a decision-making task, a user such as an editor for The Browser may wish to filter recommendations according to topics such as crime, technology, or business. The recommendations output by RFS can be controlled, by altering the per-word contributions in Equation (2) according to whether

[3] https://signal.org/

a word is topical. This is accomplished by first calculating words related to a topic word using pre-trained word embeddings from BERT [4, 7]. Words related to a topic are defined by a heuristic: the cosine similarity between all words and a topic word such as 'business' are computed, and the top 15 words closest in cosine distance are stored as topical words. Then, a slider in a visual interface is used to increase or decrease the per-word contributions of topical words to a document's logit. Let the user-input slider value be α, and the set of topical word indices be T. Then the user-controlled version of Equation (1) becomes

$$f(u, x_m) = \theta_u^\top \left(\frac{1}{|x_m|} \sum_{j \in x_m} (1 - \mathbf{I}[j \in T])\beta_j + \mathbf{I}[j \in T]\alpha \text{sgn}(w_{uj})\beta_j \right). \tag{3}$$

The sign function $\text{sgn}(\cdot)$ is applied to the per-word contribution to a document's logit. This is included since a word might contribute negatively to a document's logit, yet a user may wish to increase the weight of a related topical word.

EVALUATION

We conduct an offline empirical study of the performance of RANKFROMSETS to assess its performance as a recommendation model. Then we qualitatively evaluate the visual interface to study whether the explanation-aware, controllable interface enabled by RFS can help make editors at The Browser make better decisions.

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 results in a dictionary with 30k words, and 646k datapoints with 27k positive labels.

Metrics

Performance of the recommendation models is assessed with recall, and 15% of the datapoints are held out for the validation and test sets respectively.

Experimental setup: RankFromSets

We cross-validate using the RMSProp optimizer [6] with a momentum of 0.9 and grid search over learning rates of $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$, whether or not to initialize from pre-trained BERT embeddings [7], and embedding sizes of $\{10, 25, 50, 100, 500, 1000\}$. This model is trained on an NVIDIA Tesla P100 GPU.

Experimental setup: BERT

To fine-tune BERT, we use the AdamW optimizer with a linear learning rate scheduler and warmup steps, with a batch size of 32 and maximum input length of 512 as in Devlin et al. [4] and Wolf et al. [7]. A grid search is performed over learning rates of $\{2, 3, 4, 5\} \times 10^{-5}$, warmup steps of $\{10^2, 10^3, 10^4\}$, and total training steps of $\{10^2, 10^3, 10^4, 10^5\} \times 5$. The model is trained on an NVIDIA Tesla V100 GPU.

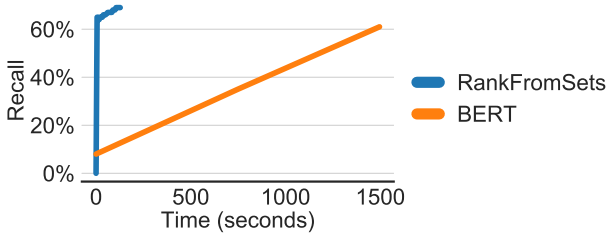

Figure 3: RANKFROMSETS achieves better performance faster than BERT in terms of validation recall during training.

Recommendation Model	Recall @ 1000 (%)
RANKFROMSETS	**53.1**
BERT	46.6

Table 1: RFS outperforms BERT in an offline evaluation, on a task of predicting which articles editors at The Browser would feature based on words in the articles.

The best-performing model of RFS is selected for deployment, and recall is evaluated on the test set, after using early stopping according to validation recall. The results are shown in Table 1, and RFS outperforms BERT by 14%. Further, RFS achieves better performance ten times faster than BERT, as shown in Figure 3. In a test to measure the speed of recommending 10^4 held-out articles, RFS ranked all articles in 120 ms on a CPU, while BERT took 4 m 54 s to rank the articles on an NVIDIA Tesla V100 GPU. This represents a 2000-fold improvement in speed, which is beneficial for the controllable visual interface that requires Equation (3) to be quickly computed in response to user input.

Qualitative Evaluation

In a user study, editors at The Browser provided feedback that they used the visual interface to choose articles, and found this to be an improved workflow. The control over recommendations, and explanation-aware visual interface provided by RFS helped elicit bugs in data collection (such as foreign language sources, or fiction writing) and provides an enjoyable user experience.

DEPLOYMENT

The visual interface is deployed on Github Pages, with the backend, RFS, deployed as a microservice on Amazon Web Services Lambda. Equation (3) is cheap to compute, so the lambda function is a short python script that requires numpy as a dependency, compared to BERT which would require a hosted GPU solution. RFS recommends recent articles from the editors' reading list of feeds. As a proof of concept, we include a tab for coronavirus-related articles that users can search through using the sliders and Equation (3).

References

[1] Jaan Altosaar. "Probabilistic Modeling of Structure in Science: Statistical Physics to Recommender Systems". PhD thesis. Princeton University, 2020.

[2] Jaan Altosaar, Wesley Tansey, and Rajesh Ranganath. "RankFromSets: Scalable Set Recommendation with Optimal Recall". In: *American Statistical Association Symposium on Data Science & Statistics* (2020).

[3] K. Cohn-Gordon et al. "A Formal Security Analysis of the Signal Messaging Protocol". In: *2017 IEEE European Symposium on Security and Privacy (EuroS P)*. 2017, pp. 451–466.

[4] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Association for Computational Linguistics*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423.

[5] Nicholas Diakopoulos. "Oxford Handbook of Ethics and AI". In: ed. by Markus Dubber, Frank Pasquale, and Sunit Das. Oxford University Press, 2020. Chap. Accountability, Transparency, and Algorithms.

[6] Tijmen Tieleman and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." In: *COURSERA: Neural Networks for Machine Learning* (2012).

[7] Thomas Wolf et al. "HuggingFace's Transformers: State-of-the-art Natural Language Processing". In: *ArXiv* abs/1910.03771 (2019).