Investigating User Expertise in Recommender Systems

Kyle Oppenheimer
Department of Computer Science
University of Toronto
kyle.oppenheimer@mail.utoronto.ca

Brendan Kolisnik
Department of Computer Science
University of Toronto
brendan.kolisnik@mail.utoronto.ca

Abstract

We propose a new metric called the 'similarityToCritic' (STC) score which measures how much a review sounds like it was written by a professional critic. This was accomplished by fine-tuning a BERT pre-trained model to distinguish between critic and non-critic reviews with 96.6% accuracy and generating probability estimates of STC for an Amazon movie reviews dataset. We then modified existing collaborative filtering algorithms to incorporate STC, and found that minor performance enhancements may be obtained for predicting ratings in recommender systems, although further validation may be required.

1 Introduction

Recommender systems are one of the most widely used applications of machine learning, and are of vital importance to companies like Amazon and Netflix to recommend new products to their users. User-based collaborative filtering (CF), a methodology often used in recommender systems, is based on the idea that we can learn a user's preferences from their past reviews, and then give them recommendations for future products based on the reviews of other users (neighbours) with similar tastes [1]. Traditional algorithms, like K-nearest-neighbours (KNN) and singular value decomposition (SVD), use numerical ratings (ie. number of stars) from a database of users, to predict a rating for an item that a user has not yet rated. Concretely, in the KNN with means equation

$$\hat{r}_{ui} = \mu_u + \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - \mu_v)}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v)}, \tag{1}$$

we predict the rating $\hat{r}_{ui}$ that will be given from a user u for an item i based on a user-mean-adjusted weighted average of the ratings r_{vi} given from each neighbour v in the neighbourhood of the k most similar users to u. Typical examples of similarity metrics $\text{sim}(u,v)$ include cosine similarity or Pearson correlation [9]. The SVD for CF equation [6] is given by

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u, \tag{2}$$

and takes a similar approach to KNN by decomposing the data into low-rank representations of the items q_i and the users p_u, as well as learning the global averages μ, user biases b_u, and item biases b_i. However, these methods do not account for the fact that not all reviews are created equal. For example, some people may put considerable thought and effort into their review and give a five star rating, while some may give little or no thought at all and still give five stars! Others will only give a rating if they either really like or really dislike an item, but not bother for anything they found

Preprint. Under review.

mediocre. Thus, it is not just the numerical rating that matters, understanding the quality of the review is potentially very useful.

1.1 Prior work

Researchers have found that identifying experts in review systems, through methods like latent-factor modelling [7], observing the adoption of domain specific language [3], and explicit feature engineering [10], can be helpful for making accurate recommendations. In "The Wisdom of the Few", Amatriain et al. [2] investigate using reviews from professional movie critics from Rotten Tomatoes to provide recommendations for general public users of a Netflix dataset. They found the K-nearest-neighbours of each Netflix user, but they only allowed the neighbours to come from the Rotten Tomatoes expert set, under the intuition that experts are more reliable and less noisy data sources.

Like "The Wisdom of the Few", we use an external Rotten Tomatoes dataset as a source of expert reviews, but we do not restrict the neighbours to come from this set. Instead, we recognize that there are varying levels of expertise in the general public dataset itself, and we can potentially identify these expert levels based on how similar a user's reviews sound compared to the reviews of known critics. Our approach was to train a model that learns to distinguish between critic reviews and general public reviews to create a new 'similarityToCritic' (STC) feature, which determines how much a review sounds like it was written by a critic from the Rotten Tomatoes set on a scale from 0 to 1.

We can now state our project objective in two steps: In step 1 we generate estimates of the STC score for each review-text as a measure of the quality of the review. In step 2 we use a user-aggregated STC score as a representation of the historical review-text of a user to try to enhance the prediction of new ratings of movies that the user has not yet seen (and therefore has no associated current review-text).

1.2 Dataset

We created a similar dataset to the one used in "The Wisdom of the Few", comprised of 110,000 reviews from the Rotten Tomatoes critic reviews dataset [2], and 110,00 reviews from the 2014 Amazon movie reviews dataset[1] [5]. Reviews from the Rotten Tomatoes dataset were labelled as 1 for 'critic' whilst reviews for the Amazon dataset were labelled as 0 for 'not critic'. We attempted to purge the datasets of any distinguishing factors to ensure that the model could only distinguish on critic level. The main example of this was the differing review lengths, as critics wrote shorter reviews capped at 250 characters, as seen in Figure 1a. To account for this we truncated all the reviews to the first 32 tokens.

A train, validation and test set were generated using an 80-10-10 split for both steps, and we evaluated using accuracy in step 1 and mean squared error (MSE) of the 1 to 5 star predictions in step 2. The step 2 dataset consists of 110,000 examples also taken from the Amazon movies dataset, but completely disjoint from the examples used in step 1. We tried to make the dataset in step 2 realistic by splitting the test set at a certain time. As our focus was not to predict reviews for new users/items, we eliminated any test users/items that did not appear in the train set.

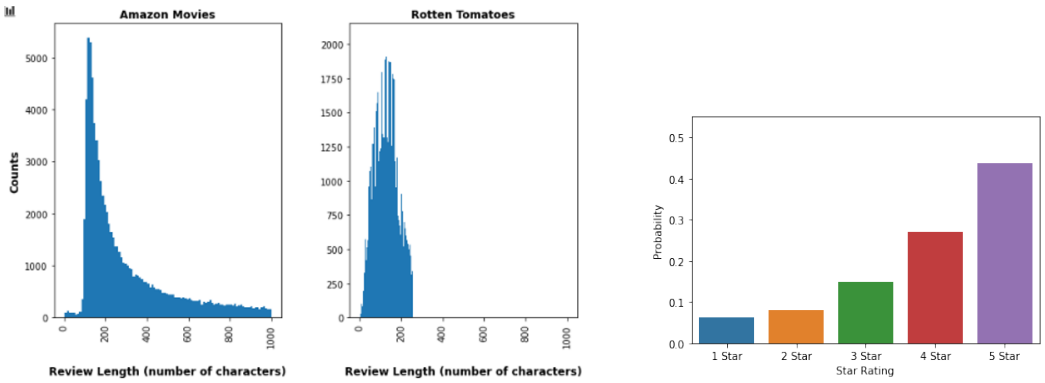
In our exploratory analysis of the dataset in step 2 we discovered that the target variable is very right skewed (Figure 1b), which is why we used MSE versus mean absolute error (another commonly used CF metric) to increase the penalty for overfitting toward higher ratings. Futhermore, the number of unique reviewers and movies is 8351 and 6692, respectively, which led to a user-item matrix sparsity coefficient of around 0.002, which is significantly more sparse than most of the papers in our literature review.

2 Model

2.1 Step 1: BERT STC predictor

For step 1 our task was to estimate the STC score for each review-text in the training data of step 2. To do this, we fine-tuned a state-of-the-art pretrained BERT model [4] to distinguish between Rotten Tomatoes and Amazon reviews. We added a final dense linear layer with two binary output classifiers

[1] We substituted out the Netflix dataset because it did not have a review-text field.



(a) (b)

Figure 1: (a) Distribution of review-text lengths for the step 1 dataset. (b) Distribution of the number of stars given for the step 2 dataset.

onto the end of the BERT network, taking the argmax as the network's prediction used for accuracy and a softmax weighted average as the continuous-valued STC score. The model was trained via supervised learning using the AdamW optimizer. After some hyperparameter tuning, this model was able to distinguish between Rotten Tomatoes and Amazon reviews with 96.6% accuracy.

The downside of using these BERT predictions is that it is hard to prove (short of manually inspecting the output) that it is distinguishing based on how much a review sounds like a critic and not by some other factor. The manual inspection we performed seemed to verify that it was working as expected. For example, many of the reviews that the model confidently believed came from the critic set began with phrases like "Vibrant, colorful, symbiotic and ambiguous...", while non-critic reviews sounded more like "This movie was very interesting! It wasn't entirely what I thought it would be...". Additionally, if the feature is successful at improving a downstream task such as rating prediction in step 2, then we can be more confident that it is extracting useful information.

2.2 Step 2: collaborative filtering with STC

For step 2 we integrated our STC score into CF models by developing custom KNN and SVD algorithms in SciPy [11], based on the following two hypotheses.

Hypothesis 1 (H1): Reviewers with higher STC scores are more reliable and therefore their ratings should have higher weighting (inspired by "The Wisdom of the Few").

Hypothesis 2 (H2): Reviewers ratings are more likely to be similar to other reviewers of approximately the same STC score. Thus, reviewer ratings should be weighted by the distance between their neighbour's STC score to their own.

To test these hypotheses, we generalized equation (1), the standard KNN with means algorithm, by adding a weighting factor related to the neighbour's STC score in equation (3), and the distance between the neighbour and user's STC score in equation (4), implementing H1 and H2, respectively:

$$\hat{r}_{ui} = \mu_u + \frac{\sum\limits_{v \in N_i^k(u)} \{\alpha \overline{e_v} + (1-\alpha) \cdot \text{sim}(u,v)\} \cdot (r_{vi} - \mu_v)}{\sum\limits_{v \in N_i^k(u)} \alpha \overline{e_v} + (1-\alpha) \cdot \text{sim}(u,v)}, \tag{3}$$

$$\hat{r}_{ui} = \mu_u + \frac{\sum\limits_{v \in N_i^k(u)} \{\alpha \cdot (1 - |\overline{e_u} - \overline{e_v}|) + (1-\alpha) \cdot \text{sim}(u,v)\} \cdot (r_{vi} - \mu_v)}{\sum\limits_{v \in N_i^k(u)} \alpha \cdot (1 - |\overline{e_u} - \overline{e_v}|) + (1-\alpha) \cdot \text{sim}(u,v)}, \tag{4}$$

where $\overline{e_v}$ is the user aggregated STC score of the neighbour, $\overline{e_u}$ is the user aggregated STC score of the user, and $\alpha \in [0,1]$ is a hyperparameter which controls the relative importance of the STC scores

versus the other similarity metric. Our most successful user aggregator was a user mean, however future work could investigate methods that better track user progress like an exponentially moving average. We also implemented an SVD analogue of H1 trained via stochastic gradient descent, by modifying equation (2) using the method proposed in the work of Song et al. [10]. To our knowledge, no direct H2 analogue exists for SVD. The results for these methods are compared to the baseline KNN and SVD algorithms in Table 1 to see if the sole addition of the STC score can significantly improve rating predictions.

We decided to use KNN because it was easy to incorporate our new feature and the results were easily interpretable. However, KNN is far from the state-of-the-art, and has the added downside of requiring us to store the entire user-item preference matrix in memory, which limited the size of the dataset we could run with our computing resources. SVD required less memory and achieved better MSE, but it is not as interpretable as KNN and is more difficult to modify to incorporate the STC score. Given these cons, we tried to implement some other state-of-the-art methods such as factorization machines [8] that could take as input user-features like the STC score, but we encountered issues with overfitting that we could not resolve given the time constraints.

3 Results

We performed a grid search over many hyperparameter values using our training and validation datasets. As seen in Table 1, STC outperforms each of the baseline results, but by a very narrow margin. This slight improvement matches the trends in the "Wisdom of the Few". Based on these results it is possible that using STC may lead to a minor performance increase over baseline CF methods for a given dataset. To validate this claim, the feature would need to be tested on additional datasets, and incorporated into state-of-the-art models such as factorization machines and neural networks to see if a more substantial lift could be attained.

Table 1: MSE results for the baseline algorithms and their STC-modified counterparts. For the SVD models λ is a hyperparameter that controls the strength of regularization, and γ is the learning rate.

Model	Top Hyperparameters	Test MSE
KNN with Means	K: 40	1.4590
KNN with Means STC H1	K: 40, α: 0.15	1.4486
KNN with Means STC H2	K: 40, α: 0.25	1.4339
SVD	Epochs: 20, γ: 0.005, λ: 0.02	1.1661
SVD STC	Epochs: 20, γ: 0.005, λ: 0.02	1.1511

4 Conclusion

In our experiments we have found that the rating performance advantage attained by using the STC score in CF models is limited. This is likely in part due to the distribution of the STC score, as the vast majority of users receive a score close to 0 by design, which limits the possible information gain. Note, the possible information that can be encoded into a single scalar variable is already limited. Future work could look into vector representations of the STC score, as well as further optimizations such as scaling by the number and frequency of a user's reviews. In addition, it would be helpful to have training labels that better signify a user's expertise, as expertise is a continuous spectrum and is unlikely to be modelled well by a single critic or not-critic label. Since this data is often not available, it would be useful to turn to unsupervised methods such as latent factor modelling of expertise. Lastly, STC may potentially be useful in an array of other applications, including expert detection or helpfulness prediction in review or Q/A communities, as well as improving the quality of writing for applications like Grammarly.

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005. doi: 10.1109/TKDE.2005.99.

[2] Xavier Amatriain, Neal Lathia, Josep Pujol, Haewoon Kwak, and Nuria Oliver. The wisdom of the few a collaborative filtering approach based on expert opinions from the web. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 532–539, 01 2009. doi: 10.1145/1571941.1572033.

[3] Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. No country for old members: User lifecycle and linguistic change in online communities. In *Proceedings of WWW*, 2013.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

[5] Ruining He and Julian J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *CoRR*, abs/1602.01585, 2016. URL `http://arxiv.org/abs/1602.01585`.

[6] Yehuda Koren. Collaborative filtering with temporal dynamics. volume 53, pages 447–456, 01 2009. doi: 10.1145/1557019.1557072.

[7] Julian J. McAuley and Jure Leskovec. From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. *CoRR*, abs/1303.4402, 2013. URL `http://arxiv.org/abs/1303.4402`.

[8] Steffen Rendle. Factorization machines. pages 995–1000, 12 2010. doi: 10.1109/ICDM.2010.127.

[9] J.L. Sanchez, Francisco Serradilla, E. Martinez, and Jesus Bobadilla. Choice of metrics used in collaborative filtering and their impact on recommender systems. pages 432 – 436, 03 2008. doi: 10.1109/DEST.2008.4635147.

[10] Sang-il Song, Sangkeun Lee, and Sang-goo Lee. Determining user expertise for improving recommendation performance. *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication, ICUIMC'12*, 02 2012. doi: 10.1145/2184751.2184833.

[11] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

5 Source Code

This project is available at the following Github link: `https://github.com/bKolisnik/CSC2515FinalProject`

6 Attributions

Both group members contributed equally. Kyle focused on step 1, BERT STC predictions. Brendan focused on step 2, KNN and SVD implementations. Both members contributed to data processing, literature review, and deliverables.