Adapting Transparent Recommendation Model for Explainable Candidate Evaluation

Ethan Eldridge
CSCI-B659 Final Project, Indiana University
etmeldr@iu.edu

Abstract

Candidate evaluation is a necessary but tedious process for career and scholarship opportunities. Applications for such opportunities are often laden with dense, but relevant data. Without taking proper organizational and record-keeping measures, this already-profuse data can easily be lost among the numerous candidates applying for the opportunity. This project adapts an existing model from Balog et al. in order to model evaluators' interests and recommend them candidates that fit their interests based on the given data. In this way, evaluators can simply define their interests for the model, and the model will keep track of all candidates' relevance in a way that a human could not on the scale of hundreds or thousands of candidates.

Introduction

Candidate evaluation is a necessary but tedious process for career and scholarship opportunities. Applications for such opportunities are often laden with dense, but relevant data. Without taking proper organizational and record-keeping measures, this already-profuse data can easily be lost among the numerous candidates applying for the opportunity. Most evaluators fail to address the unwieldy nature of this data and end up choosing candidates based on abstract or ill-defined criteria. Although each candidate should have the chance to be evaluated holistically and without consideration of bias or other external factors, the slog of evaluating hundreds or thousands of candidates can prevent even the most conscientious evaluators from keeping track of all the relevant data a candidate has to offer. It then follows that "evaluator fatigue" can interfere in the fairness of candidate evaluation. In order to mitigate gaps in an evaluator's understanding of a candidate, it would make sense to take detailed notes on reach candidate for reference during discussion. After all, it would be impossible for one evaluator to keep track of each and every data point, such as GPA, test scores, essay quality, financial need, etc., over thousands of candidates, without keeping a written record of this information. This note-taking process would undoubtedly be very time-consuming, however, and aside from having the notes available as a static reference for individual candidates, the notes would not be drastically improving the speed or efficiency of the evaluation process.

This project adapts an existing method from Balog and colleagues to address the problems often associated with candidate evaluation. The method, from the paper "Transparent, Scrutable, and Explainable User Models for Personalized Recommendation," was introduced in 2019 to improve the transparency and explainability of content recommender systems common in services like Netflix, Amazon, and others (Balog et al. 2019). The Transparent, Scrutable, and Explainable (TSE) method makes use of user-defined "social" tags on each item to create a model of the user's interests. The method then takes user feedback on the information it has gathered so far to improve the functioning of the model and build trust with the user. Finally, using the weights the model has calculated, each item is ranked and recommended to the user according to how likely the item is to match the user's interests. My project adapts this content-recommendation system for use in candidate evaluation.

Motivation for Project

In 2004, my mother passed away after a short battle with lung cancer. Friends of my family started a scholarship fund in memory of my mother, and each year since the fund was started, my family has raised money to give out two scholarships annually to local high school seniors. I recently joined the board for the scholarship fund and became intimately familiar with the tedious process of candidate evaluation during this year's selection process. Though our scholarship only had 37 candidates this year, it took each member on average a little over 2 hours to read through each application. Additionally, when it came time to discuss which candidates were going to receive the scholarship, I found several inconsistencies between the stated interests of the selection committee and the candidates that were recommended. For example, before starting the selection process, our committee decided that, amongst other aspects of a student's application, we would prioritize students who had less money to pay for college over those who had already received other scholarships, or whose families had a higher estimated contribution based on high income. In the final recommendations from each member, however, there were several candidates that were in the lowest category of financial need (had received the most money to pay for college), and in the

lowest category for other metrics such as GPA or essay quality. When asked about these discrepancies, the other members of the board admitted that they had lost track of these data points over the course of evaluating the 37 candidates that applied, and had simply chosen the final recommendations based on "overall feeling." Given that the applicant pool for our scholarship is small relative to other scholarships or job postings, and that our committee was already forgetting some candidates' data, I would imagine that this problem is likely much worse for opportunities that have a large applicant pool.

After having had some experience working through the candidate evaluation process, I thought it would be helpful to adapt the method from the Balog paper to solve a few of the problems my committee was having. The TSE method is particularly well-suited to the problem of candidate evaluation, as it uses user-defined "social tags" to create a model of the user's interests. The model was originally created for content recommender systems, but consumers of movies or other media rarely need to or are able to explicitly define all of their preferences before consuming content. On the other hand, selection committees often have their ideal criteria explicitly defined before starting the candidate evaluation process, which I thought would integrate well with an explicitly defined tag-based system. Additionally, the explainability and transparency of the model would allow the evaluators to keep track of how and why the model makes its decisions to ensure that it has a correct understanding of the evaluator's preferences.

Related Work

The first and most heavily related reference for this project was Balog et al.'s paper, "Transparent, Scrutable and Explainable User Models for Personalized Recommendation." At the time of submission, the project is almost functionally equivalent to the model outlined in this paper. Given that this project is heavily based on Balog et al.'s paper, many of the works related to the TSE model would also be related to this project. This project and the TSE model are content-based, meaning that they make connections between user-defined ratings and other content-based information to match the user with items.

Related, but different models are collaborative filtering-based models, which use information about users with similar interests to the primary user's to recommend items. Elements of collaborative filtering methods, such as those detailed in references 2-4, can be added to the TSE model and this project to make a hybrid method of sorts, but this is not necessary for the functioning of the TSE model and is not the primary way by which the model makes decisions. Other related works differ in the way that they present information to the user. In the TSE model, for example, information that the model has gathered about user preferences is handed back to them in "natural language" format. Similar content based methods instead use formats such as word clouds to present information in an explainable manner [5].

Finally, it is worth noting that there have been a few other attempts at solving the candidate recommendation problem, though none of them in exactly the same way as this project. In reference 6, for example, Roy et al. uses a Linear SVM and different text-extraction models to take resume information and use it to recommend candidates to a user. This model differs from the TSE model and this project in that it does not have an explainable component to it, and therefore offers no additional information to the user to help facilitate understanding, nor does it invite additional feedback from the user to correct the model's understanding.

Notes on Construction of Project

I began working on this project by reading through Balog's paper on the TSE model and taking notes on the set-based and statistical mechanisms behind the system. Given that I do not have much experience in statistics of the kind presented in the Balog paper, I took a great deal of time to go through the calculations, both on my own and with Prof. Leake, in order to better understand how they would be implemented in the final program. After reading through the paper and annotating my findings, I began to code the program. Given that the details of the program's functioning are supposed to remain high-level in this paper, I will not write the equations for the calculations of any metrics in this paper. Instead I recommend looking at the Balog paper in reference 1 of the bibliography.

The overall flow of the program is largely the same as what is laid out in the Balog paper, with some slight modifications. In the Balog paper, the specifics of the process used to collect item ratings and tags is not very explicit. To account for this in my program, I created a routine that uses a preexisting candidate database and allows the user to choose which candidates from the database he or she would like to rate and tag. Not all candidates need to be rated or tagged, and the program keeps track of which candidates have been processed by the user so that the other candidates do not interfere with the model's functioning. Instead of immediately rating candidates on a scale of [-1, 1] as in the Balog paper, candidates in this project are rated on a scale of [0-10] given that most users I have in mind would be more familiar with this scale. The [-1, 1] is useful, however, as negative numbers indicate a negative preference for an item, so each [0-10] rating is immediately converted back to the [-1,1] scale for use in the program.

After creating the data-collection routine and documenting each candidate's rating and tag, the program begins to calculate tag-level ratings, weights, and utilities for each of the tags. The tags also undergo the Wilcoxon Signed Rank test in order to verify a preference that is statistically significant from zero. Following the processing of the single tags given by the user, pairwise tags are created by the program, and ratings, weights, and utilities are also calculated for these tags.

After ratings, weights, and utilities have been calculated for all tags (single and pairwise), the top k tags by utility are chosen to be converted to natural-language statements and presented to the user. These statements serve as representations of the model's understanding of the user's preferences, so it is important that the information provided to the user is as clear and understandable as possible. To facilitate the conversion of a tag to a natural-language statement, a func-

tion, $naturalize()$, was created. It takes in a tag (single or pairwise) and its tag ratings and returns a natural-language statement summarizing the user's preferences for candidates with that tag. These statements fall into one of 6 categories, all of which have "like" or "dislike" variations depending on the rating of the tag. Depending on the absolute value of the user's rating for the item, each statement can also be qualified as "strongly" like/dislike (ratings of 7.5/10 or above; 0.5 on [-1,1] scale or above). This feature was included in the potential improvements of the TSE model listed in the Balog paper. Most of these categories were presented in the Balog paper, but I added statement #6 below to account for pairwise tags that identified no preference changes between two single tags:

1. Single – "You (dis)like candidates tagged with {tag}."
2. Especially If – "You (dis)like candidates tagged with {tag1}, especially if they are also tagged as {tag2}."
3. Especially If Not – "You (dis)like candidates tagged as {tag1}, especially if they are not also tagged as {tag2}."
4. Unless – "You (dis)like candidates tagged as {tag1} unless they are also tagged as {tag2}."
5. Exist (as in, only if tag2 exists) – "You only (dis)like candidates tagged as {tag1} if the candidate is also tagged as {tag2}."
6. Indifferent – "Your opinion of a candidate tagged as {tag1} does not seem to change when that candidate is also tagged as {tag2}."

Figure 1 below is taken from the Balog paper and offers a visualization of most of these categories as they relate to the natural-language statement they are translated into:

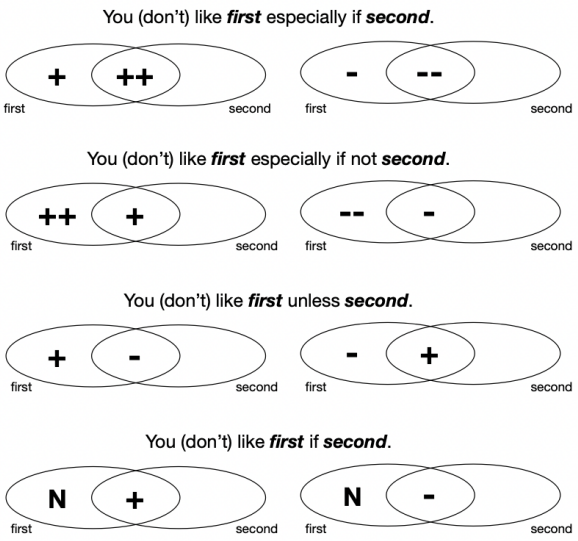

Figure 1: Venn diagrams illustrating the user preferences for pairwise tags that align with each natural-language statement, where "First" = tag1 and "Second" = tag2

In addition to the natural-language statement generated from the tag's metrics, an example from the user's rated candidates is selected to be presented alongside the statement. This candidate fits one or more of the tags implicated in the statement and is chosen based on the content of the statement. For example, although statements from the Especially If and Especially If Not tag categories both involve pairwise tags, the statement "You like candidates tagged with 'good GPA,' especially if they are also tagged as 'college graduate'" would require an example tagged as both "good GPA" AND "college graduate." On the other hand, a statement from the Especially If Not category, such as "You like candidates tagged with 'good GPA,' especially if they are not also tagged as 'criminal record'" would require an example that is tagged with "good GPA" and not with "criminal record." Additionally, given that a tag's rating is an average of the ratings of the candidates using that tag, not all of the candidates for that tag will match the content of the statement (ex. although the majority of candidates tagged with "good GPA" have positive ratings, some may also be tagged with "criminal record" and therefore have bad ratings). Instead of choosing a random candidate underneath a given tag, I quickly found that I had to be careful to select an example from the tag that matched the content of the natural language statement. Very little of this, if anything, was detailed in the Balog paper, so I was forced to devise and implement my own methods for solving this problem. For these reasons, a great deal of attention was put into the logic of the $naturalize()$ function and its complementary example selection function.

Once each of the top-k statements and supporting examples has been presented to the user, the user likelihood for each candidate is calculated and the candidates are recommended to the user in descending order of likelihood. No further actions are taken until more candidates are added to the database, or until the user indicates that he or she would like to make changes to the weights of existing candidates.

Experimental Methods

In order to test the performance of the model, I decided to generate 30 fake candidates (approximately equal to the number of candidates I recently had to evaluate for my scholarship committee). I randomly assigned each of the 30 candidates into three groups: strongly liked, liked, and disliked. Within these three groups, ratings were randomly assigned according to the specified preference for each item (ex. strongly liked tags were given a random rating between 0.6 and 1 on the [-1,1] scale), and tags were randomly assigned to portions of each group to ensure that the following interests were consistent:

- I like candidates tagged as "Good GPA"
- I dislike candidates tagged as "Bad GPA"
- I like candidates tagged as "Good GPA," especially if they are also tagged as "Grad Degree."
- I like candidates tagged as "From Indiana," especially if they are not also tagged as "Purdue Grad."
- I do not like candidates tagged as "Little Work Experience," unless they are also tagged as "Volunteer."

An illustration of the distribution of tags amongst the candidates in the experiment can be found in Figure 2 below.

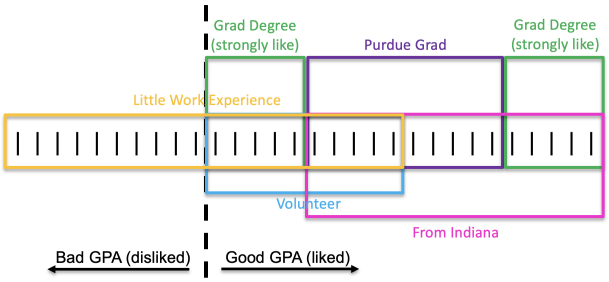

Figure 2: Illustration of the distribution of tags amongst the 30 candidates of the experiment

It was hypothesized that the interests laid out earlier in this section would be identified by the model, and that they would appear in the top-k statements returned by the model. It was also hypothesized that the candidates with the highest user likelihood would be those tagged "Grad Degree," and that they would appear at the top of the recommendation list.

Results

The results of the experiment can be seen in Appendix 2 at the end of the paper. Appendix 1 is meant to demonstrate the full functionality of the model, so it has a smaller candidate database than Appendix 2 does. Appendix 2 intentionally does not contain examples of the rating and tagging process for each candidate because there would be 30 candidates in total to rate and tag, which would be a very lengthy output.

Discussion

Discussion of Results

From the output of the code shown in Appendix 2, some interesting observations can be taken. Firstly, the model was at least able to pick up on on the preference for candidates with a graduate degree; the top 4 candidate recommendations held graduate degrees, and the top statement returned in the summary of user interests was "You strongly like candidates tagged with "Grad degree." As for the other preferences, however, the results were not as clear. In the statement "You like candidates tagged as 'Little work experience,'" for example, the program was not necessarily wrong, but it wasn't what I had intended the output to be. Given that the strong like:like:dislike ratio of candidates with little work experience is 25:25:50, I can see where the program might have thought that I show a preference for candidates with the single tag "little work experience." I did technically like 50 percent of the candidates tagged with "little work experience" (and 25 percent of those I strongly liked), but in my stated preferences before the experiment, I was trying to express that I did not like candidates labeled as such unless they were also volunteers. The statement closest to what I had stated in my interests was "You like candidates tagged as 'Little work experience,' especially if also tagged as 'Volunteer,' but it was also not worded such that it clearly expressed my preference, and it did not have a utility greater than 0, so I am not confident that the model picked up on this interest. More work would likely need to be done to compare the content of the top-k statements before they are presented to the user in order to avoid any contradictory or redundant information.

This example, among a handful of others, does not have a very clear explanation. Although I rigorously tested the calculations in the model, it is possible that an error could have been made in the calculations or setup of the model as I understood it. Given the page limit for research papers, Balog et al. did not include very much in the way of specific details in their summary of the TSE model, so I was forced to infer at certain points of the model construction.

Another idea is that the model did not have sample sizes large enough to identify the preferences stated earlier. Many of the calculations performed in the model are based on statistical models that hold specific assumptions about the distribution and size of the candidate pool. It is possible that the calculations for many of the tags were inaccurate for this reason, especially in the case of pairwise tags which are only populated with candidates that overlap between tags. It is worth noting that the original TSE model got its results after being tested on a publicly available dataset of more than 17 million ratings, 100,000 users, 5000 items, and 5000 tags, while my project only tested on 30 ratings, 30 candidates, one user and seven tags. An inability to obtain accurate results on a small dataset would be a major weakness of my project given that the candidate pool for my and many other community scholarships does not reach into the thousands or even hundreds of candidates.

Additionally, it is possible that my experimental design was errant. There was a great deal of overlap between the tags (for example, the "little work experience" result detailed earlier), and it is possible that my interests were not as clearly stated as I had hoped in my distribution of the tags.

Finally, although I did improve upon Balog et al.'s model in some regards (adding a preference strength qualifier, for example) there are a few elements of the original model that I did not include here. One important element, for example, is the candidate prior calculation drawn from collaborative filtering. In the results of the Balog paper, it is shown that, although the priors are optional, the performance of the model was significantly better using the collaborative filtering priors than when the model did not incorporate them into the user likelihood calculations. I did not include these in my model, and it would be one of the first components I would add to this project given more time. Additionally, there was a "smoothing parameter," μ, that was used in the user likelihood calculations of the Balog paper. Because it was not defined in the paper, I assume that it is something to be tuned and not a universal constant. Tuning the smoothing parameter was outside the scope of this project, however, and I did not have the time to test it given that the construction and proper operation of the model was my priority. Instead, I set it to a constant value of 1, which could have had some effect

on the results of the experiment.

Suggestions for Future Work

Given that I am a group of one, and that the project in its current state took a great deal of time to complete, there is still a good amount of improvement that could be made to this model. Aside from collaborative filtering priors and a tuned smoothing parameter, I would have liked more time to build an interactive step in the workflow of the program for the user to be able to modify the weights of the tags and of the statements that are presented. This was a crucial part of the experimental design of the TSE model, and the results of the experiment improved with user input, which is something my project did not have. I would have also liked to add a graphical user interface and a counterfactual explanation system so that users could ask questions about the rankings of different tags and items to alleviate any confusion and better understand the model. Finally, I would also like to have tested my model on a much larger dataset in order to compare the results. Given the time constraints, however, this was not possible.

Conclusion

In conclusion, this project was a fantastic learning experience that allowed me to better understand the principles of explainable artificial intelligence and how to apply them to a real-world scenario. Although, given the experimental results, this model may not be the perfect solution to the specific use case I am looking to satisfy, many of the ideas and mechanisms used in the Balog TSE model will be useful in the final product I develop for my scholarship committee.

References

1. Krisztian Balog, Filip Radlinski, and Shushan Arakelyan. 2019. Transparent, Scrutable and Explainable User Models for Personalized Recommendation. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19). Association for Computing Machinery, New York, NY, USA, 265–274. https://doi.org/10.1145/3331184.3331211

2. Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. 2000. Explaining Collaborative Filtering Recommendations. In Proc. of CSCW'00. 241–250.

3. Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In Proc. of ICDM'08. 263–272.

4. Yehuda Koren and Robert Bell. 2015. Advances in Collaborative Filtering. In Recommender Systems Handbook (2nd ed.), Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). Springer US, Chapter 3, 77–118.

5. YongfengZhang.2015.Incorporating Phrase-level Sentiment Analysis on Textual Reviews for Personalized Recommendation. In Proc. of WSDM'15. 435–440.

6. Pradeep Kumar Roy, Sarabjeet Singh Chowdhary, Rocky Bhatia, A Machine Learning approach for automation of Resume Recommendation system, Procedia Computer Science, Volume 167, 2020, Pages 2318-2327, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.03.284.