ForumRec - A Question Recommender for the Super User Community

Data Science Capstone

Yo Jeremijenko-Conley
Jasraj Johl
Jack Lin

The Super User forum exists on the internet as a medium for users to exchange information. In particular, the information shared here primarily related to questions pertaining to operating systems. The system we developed, ForumRec, aims to increase usability for the forum's participants by specifically recommending questions that may be more suitable for a user in particular to answer. The model we built uses a combination of technique content-based and collaborative filtering from the LightFM package to identify how well a novel question would fit for the desired user. In comparison to baseline models of how Super User already recommends questions, the model attains better performance for more recent data, scoring 0.0014, 0.0033, and 0.5160 on precision at 100, recall at 100, and AUC, which is markedly better than the baselines.

Introduction

The goal of a forum such as Super User [1] is to assist users to the best of their ability in finding help, or providing help, for whichever topic they may have in mind. The website hosts a vast number of queries, with a range of topics from Ubuntu to Windows-10. For instance, topics about “command-line” might include anything about interacting with the computer that solely uses a textual environment, as opposed to a graphical one. The website interface is introduced with a list of questions, as well as sections marked by Questions, Tags, Users, and Unanswered. Additionally, it is also possible to sort by Active, by Bountied, or by question age.

However, the forum currently does not provide certain functionalities tailored towards unanswered questions, which are plenty on this forum. When selecting the Unanswered section, users are able to sort by the user’s preselected tags, newest, votes, or questions with no answers. While the preselected tags might offer a good starting point, each field however is still too large that one user’s expertise may not cover the entirety of the given tag. While a user may be overly familiar with the current version of Mac OS, they may not even have user previous versions of Mac OS, which could be a different architecture altogether, yet still falling under the same tag.

One of the biggest challenges for engaging new users however is the retention rate of new users. Existing users to Super User might already have some semblance of an idea as to where new questions can be answered, but novel users will have zero idea as to where to start. ForumRec primarily aims to provide a better starting platform for new users, or “Cold Start” users. Users who have either never answered any questions or have only utilized Super User a handful of times can log on to ForumRec to be provided with a sampled list of questions. The user then only has to indicate whether or not they think they could answer each of these questions instead of actually answering them. This saves the user time in prepping the user’s account for our model, instead steering the user’s interest and expertise level towards the sampled questions. These indications are used as the user’s input instead of previously answered questions, whereby recommendations are then provided to the user.

To better recommend questions for users to answer, analysis of the content needed to be done. We used content-based and collaborative filtering in a hybrid model using the LightFM package to determine the interactions and relationships between users and certain questions in the data. Natural Language Processing (NLP) with a Term Frequency - Inverse Document Frequency (TF-IDF) matrix were used to parse the given textual body helping the model provide insight towards which words may carry more significance due to lowered frequency, or use in specific situations with other words. Along with a one-hot encoding of the tag data, we were able to contrast a model

to give recommendations to users based on their question history. So, for a given we are able to find the questions they are most likely to answer.

For the models that we have, we need to compare them against baseline to ensure that the work we are doing is effective. To do this, we used the strictly collaborative filtering baseline and were able to get a 51.6% AUC score compared to the baseline's score of 50.01%. We also see scores of 0.14% and 0.33% for the model compared to the baseline's scores of 0.10% and 0.27% for precision and recall at 100 respectively.

Data

The data used for this recommender system was gathered by archive.org and is from the Super User forum of Stack Exchange. The dataset contained information on posts, post history and changes, post links, comments, badges, tags, users, and votes, however, for this recommender system, we used a subset of this data. We used posts themselves as the main source of data. In posts themselves, we maintained the id of the post, the post type (question or answer), the creation date of the post, the score of the post (up votes and down votes given to the post), the text of the post's body, the tags associated with the post, the id of the owner of the post, the id of the parent post (only used for answers to determine the question they are associate with), and the number of answers for that post. With these pieces of information we believed we could create the recommendation system we were striving for. We could use the text of the body to do NLP based analyses, use the tags to categorize and potentially find similar posts in space. We would then make sure to associate the questions people have answered, their answers, and partially the score of their answers to them to be able to determine which questions they would be best to answer. The recommendation system would be entirely based on these few aspects of the data, however, with a few gigabytes of data, we believe this is enough to build this recommendation system, as we have enough data per most people to be able to get recommendations. [2]

To get more relevant data for our model to be accurate on recent data inputs, we split the data to focus on more recent posts. All the data had the attributes listed in the previous paragraph and was using the posts data. The data was split on January 1, 2018 to get the last three years of data. We also made sure all posts in the dataset were questions that were answered, or the answers to those particular questions. We also include users in the dataset that answered at least 25 questions in the past only. This was to ensure that the model would be able to get personalized recommendations for users as we needed large amounts of data on questions they have previously answered to build more robust and diverse predictions. We did previously have training and testing data split on the last year 3 months of data when we attempted to use a cosine similarity model with NLP, but as we went away from that we no longer split the data into training and testing data. We instead shifted and are now creating a hybrid collaborative filtering and content-based filtering model to create our interactions and get predictions, so we needed to use all the data to create our matrix and did not split the data into training and testing data sets as the model does that itself. We would instead measure how our predictions would result from within the matrix that we build between questions and users. This gives us over 64000 questions as data points over 110000 answers as data points for our model.

Lastly, we also are gathering data from the Super User itself using the Stack Exchange API [3]. We are able to gather potentially 10000 data points daily from stack

exchange itself, and this data is being used for the user interface of our recommendation system. By being able to collect data from Super User on a consistent basis, we can be able to consistently update users with questions relevant to them through our front end interface as it is occurring. This data is filtered to only the Super User forum, and can be filtered on date time to ensure we do not collect posts already collected. The filter also has the score, tags, creation date in unix time, post type (question or answer), title and body of the data available for us through the API, and this is all that is needed to reflect the data given from archive.org's dataset. We have taken data from December of 2020 to the beginning of March and added that data to our model in order to keep our data up to date. We then update our data every 3 days to keep our model updated and provide recommendations for newer questions. We can also update the data more quickly with a shorter period of time to update on to get better and more relevant results.

Methodology

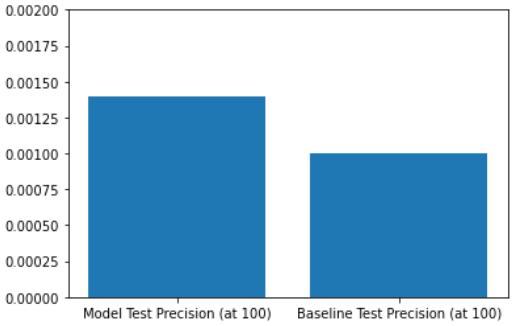
In order to recommend these unanswered questions to users, we initially built a recommender that uses linear kernel similarity metrics between questions. The body text of each question in the training data was tokenized, building a large TFIDF matrix. This model would take in the index of a forum question, it would then use the TFIDF matrix to compute the linear kernel similarity scores of the forum question against all other questions. They would then be sorted and cut down to the top 100 most similar questions, and the model would then return the set of users who have answered one of these 100 questions. However, this model had many shortcomings, while being based entirely on content based filtering, it had no means to optimize the weights of the feature matrices, and was very computationally expensive when making recommendations. We decided to switch to using the LightFM package to create our model, LightFM gives the flexibility to create a hybrid model which utilizes both content based and collaborative filtering. To use this the data had to first be reformatted to work with the LightFM model. We created an interactions matrix between users and forum questions, each interaction represented a question (corresponding to it's column) which was answered by a user (corresponding to its row), these interactions were weighted by the answer's score (# of upvotes - # of downvotes). The TF-IDF feature matrix was already in the proper format, however we created an indice for each forum question which mapped their IDs to their corresponding row in the feature matrix matrix. We also created additional item features by expanding the column of the tags attached to each question into a matrix of binary variables for each tag (1 indicating that the tag was used in the question, and 0 otherwise).

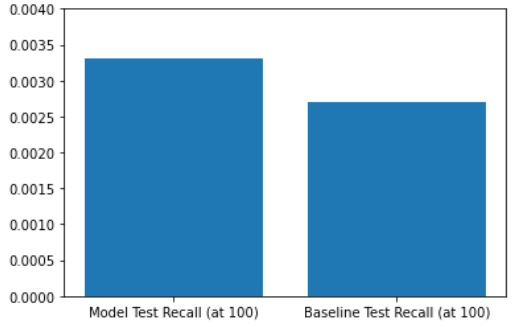
When fitting the model (during development) the interactions matrix was separated into training and test data with a 70:30 split. This split maintained the structure of the matrix (keeping the same users and questions for both), but split the interaction values in the matrix. This meant the training data kept 70% of the non-zero matrix values (the rest being changed to zero) while the test data kept the 30% omitted from the training data. This means that the feature matrix did not have to be split, as the model incorporates all of the questions into both interaction splits (although it may not utilize them if all interaction instances of a question were omitted from the training data). When given the training data, The model fits embeddings for users and items in a way that represents user preferences in questions over items. When the embeddings for a user and an item are multiplied together, they produce item-user pair scores, which represent the likelihood that the user is willing/capable of answering the specific question. These embeddings are learned through stochastic gradient descent, a method to minimize an objective loss function by updating the parameters (feature weights in our case) in the opposite direction of the gradient (slope) of the objective function. Being the most popular choice in recommender systems, we chose WARP as our objective

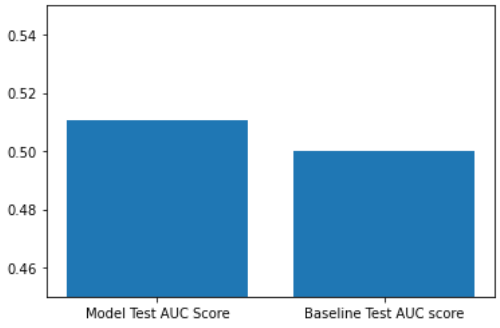
loss function. WARP stands for Weighted Approximate-Rank Pairwise loss, it optimizes the rank of positive examples by repeatedly sampling negative examples until a rank violating one is found. It is generally used when only positive interactions are present, thus given the sparsity of the negative examples in our data, it was considered the best choice of loss function.

Analysis

To evaluate the effectiveness of our recommender, specifically the effectiveness of it's content based filtering, we built a baseline model that uses only collaborative filtering in making its recommendations. This means that the baseline model was fitted to the interactions matrix, but was not given the item features which are needed for content based filtering. Despite neither model giving high values on the evaluation metrics due to the sparsity and scale of the interactions, our hybrid model out-performed the baseline on all evaluation metrics. The hybrid model achieved a precision at 100 score of 0.0014 on test data, while the baseline scored 0.0010, precision at 100 represents the average fractions of known positives (questions answered by the user) in their top 100 recommendations, meaning that there were on average 0.14 and 0.10 known positives in a users top 100 recommendations for the hybrid and baseline model respectively. While these numbers may seem very small, there are over 40,000 forum questions in the interactions matrix, with roughly 10-20 known positives per user on average, meaning that a recommender working completely at random would have a far lower score. Our next evaluation metric was recall at 100, the number of known positives in a users top 100 recommendations divided by their total number of known positives. The hybrid model achieved an average recall at 100 score of 0.0033 while the baseline scored 0.0027, again while these numbers seem small, a random recommender (which would randomly sample 100 questions out of the 41,524 in the data) would have an average recall at 100 equal to $100 \div 41,524 = 0.0024$. Our final evaluation metric was the AUC score, which is the







probability that a randomly chosen positive example is ranked higher than a random negative (or null) value, with the hybrid model scoring 0.5160 and the baseline scoring 0.5001.

Website

For ForumRec, we decided to output our recommender system onto a website. Our website is jackzlin.com [4]. It was hosted using Heroku [5] and we used an S3 Bucket from AWS [6] to manage our data and our model.

Handling of user input utilized a conjunction of services. Initially, the user loads the website, and signs in using either a new or existing Stack Overflow or Super User account [A1]. The login uses StackApps API to handle user authentication, which returns user credentials for the website to use. Upon return, the user's id is then checked internally to determine whether or not the user meets the requirements to be given recommendations for questions to answer, or if their account does not meet the threshold of answered questions and are therefore deemed a "Cold Start" user.

For a "Cold Start" user, the assumption is that the user has not answered enough questions for the recommender to accurately predict which questions would be suitable for the user to answer. Instead of the immense effort of requiring the user to go and answer enough questions on the forum, we instead offer the user a list of popular questions, where the user would indicate whether or not they think they could answer these popular questions [A3]. These popular questions were part of the training data, and the question list is pulled from a PostgreSQL table stored onto AWS' Relational Database (RDS). The output from the user is then converted into CSV format and stored onto AWS' S3.

This "Cold Start" problem brings a problem in re-fitting the model. It is possible to make a new interaction matrix that includes these new users as well as the users which the initial model was trained on and then train an entirely new model, however, this is not computationally efficient to do for every new "cold start" user that signs up. Feeding these recommendations directly into the model, or simply appending new user rows to the training data is not viable either since the model is fitted specifically to the dimensions of the interaction matrix. The model learns embeddings for each user and item through matrix factorization (from collaborative filtering), this means that the model can not update itself (partially fit) on an interaction matrix of a different size than the initial one. We solve this problem by creating dummy user rows and item columns in the initial interaction matrix. These dummy users serve as placeholders for new users and have no interactions thus do not affect the collaborative filtering process of fitting the model. New users are then mapped to these dummy indices, and their response to the list of popular questions is used to create a new interactions matrix. This new interactions matrix only contains the interactions of the new user (about 10 non zero values) but maintains the same dimensions of the original matrix. This allows the model to be partially fit to these new users (allowing the model to effectively give them

recommendations), without the need to refit the model for each user in the training data, and is thus very computationally cheap.

Finally, a script is run on Heroku to generate recommendations from a stored pickle file, which then returns a list of questions for the user [A2].

For the design of our website, we used HTML, JavaScript, and CSS, however, this CSS and JavaScript was supported mostly by Bootstrap [7]. Bootstrap is a framework to quickly design and customize responsive websites for any project using CSS, JQuery, and JavaScript. To quickly build our website, we used Bootstrap's jsDelivr functionality to embed bootstrap into our HTML and develop the design of the website using the classes in the CSS. The design for the CSS that we used came from Bootswatch [8]. Bootswatch has many design templates available to use, but the one used for this was the Lux theme made by Thomas Park [9]. By using Bootstrap, we were able to convert our HTML layout into a specific simple design with clickable links to Super User questions, a navigation bar, a contact form, a scrollable window inside a form to submit cold start questions that a user might be able to handle. Bootstrap makes it easy to turn the tags that were already made in our HTML document into specific designs that are adjustable to what is needed for the website using their class names.

Next Steps/Future Plans/Possible Future

There are many potential avenues we can take ForumRec as a recommendation system and product in future steps. One potential update would be to make the update period for the model smaller. Instead of the model being updated every 3 days, we can work to make the model update every day or potentially multiple times a day, so whenever users login to get their recommendations they can have models that better reflect the changes that are being made to Super User continuously. This would allow users to get fresher recommendations and may encourage them to look for recommendations more frequently or interact with Super User more.

Another potential update would be to get user feedback. As a product, we want ForumRec to be helping our user base and creating recommendations for them that are relevant to their expertise and also encourages them to engage with the Super User forum. By utilizing user feedback we can see how ForumRec is able to help users, and what potential changes would be needed to our model or our predictions in order to encourage users to use our platform and answer more questions on the forum site.

A lot of the work we are doing also focuses on newer users who have less experience using Super User and have not answered many questions, however, we also want to focus on more veteran users who answer questions on Super User more often. They make up a large portion of Super User's questions and answers and can have a larger impact on how ForumRec can impact Super User. We want to be able to provide them with recommendations that can help ease their experience with the forum and allow them to check our recommendations and quickly determine the questions best for them. We hope by focusing on more experienced users, we can also improve the efficiency of the forum site and allow less questions to remain unanswered on the website as we help veteran users be more efficient in their use of Super User.

Conclusion

Through a forum based recommendation system, we believe that we can provide users with an easier experience in engaging with forums like Super User. Through our recommender system, we want to be able to determine the best users to answer certain questions when a question is asked on the forum, and then send those users those questions to help them engage with the platform. Through the hybrid collaborative and content-based filtering model we have established with the posts and the users, we attempt to make that exchange, with results that are clearly better than the baseline recommendations. However, there is a lot we can do to improve the model and the precision, recall, and AUC scores we get from it. This is something we plan to improve and work on to ensure an effective recommendation system and will use the baselines and previous models to measure the performance and effectiveness of our model. In conjunction with our website, we hope to provide users a satisfying and encouraging experience with ForumRec that helps improve the efficiency and unanswered questions problem within Super User.

Citations/References

[1] Super User Forum. https://superuser.com/

[2] Abel F., Bittencourt I.I., Henze N., Krause D., Vassileva J. (2008) A Rule-Based Recommender System for Online Discussion Forums. In: Nejdl W., Kay J., Pu P., Herder E. (eds) Adaptive Hypermedia and Adaptive Web-Based Systems. AH 2008. Lecture Notes in Computer Science, vol 5149. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-70987-9_4

[3] Stack Exchange API. https://api.stackexchange.com/

[4] ForumRec Website. https://jackzlin.com OR https://forum-rec-app.herokuapp.com

[5] Heroku. https://www.heroku.com/

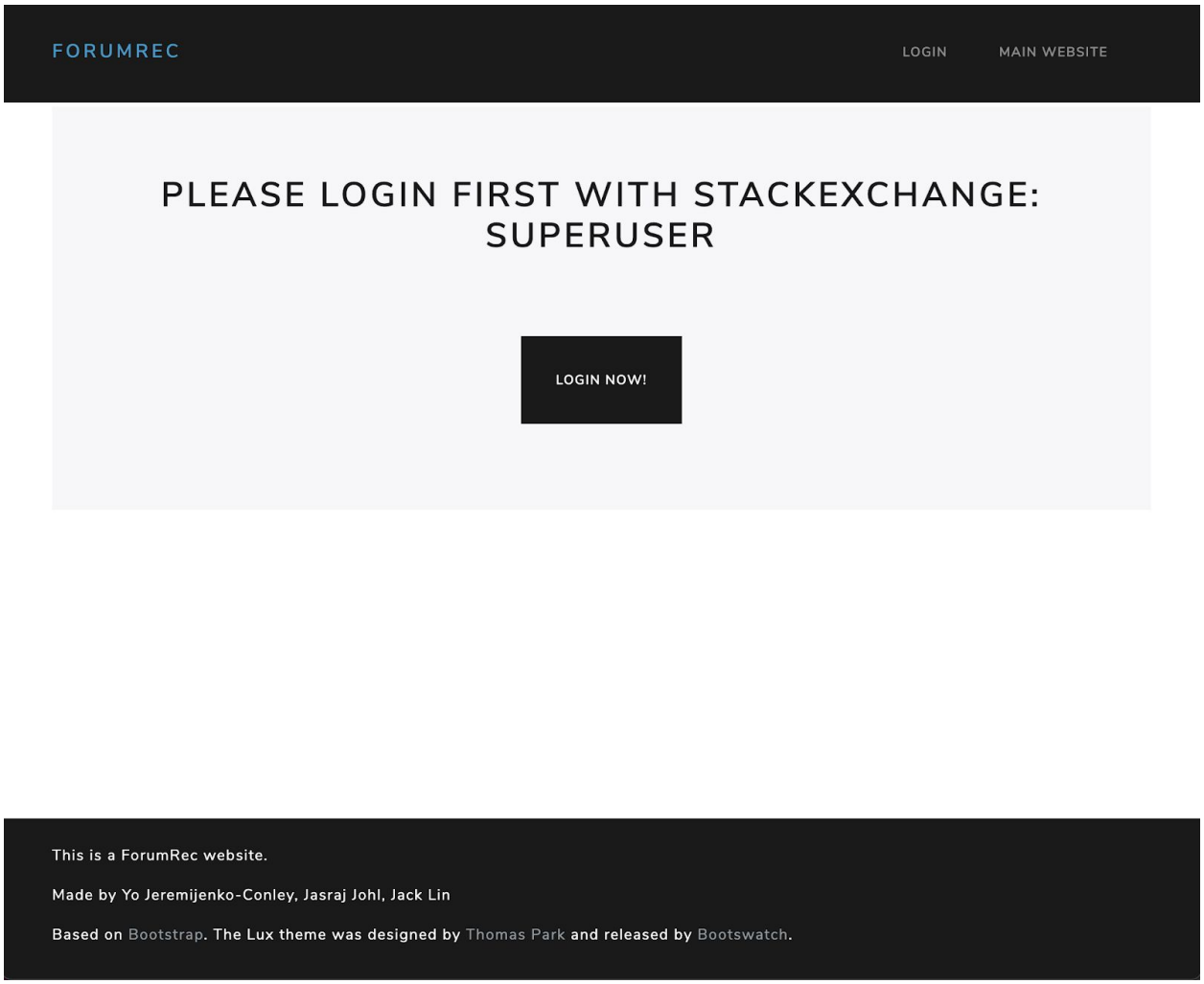
[6] AWS (S3, AWS) https://www.aws.amazon.com/

[7] Bootstrap. https://getbootstrap.com/

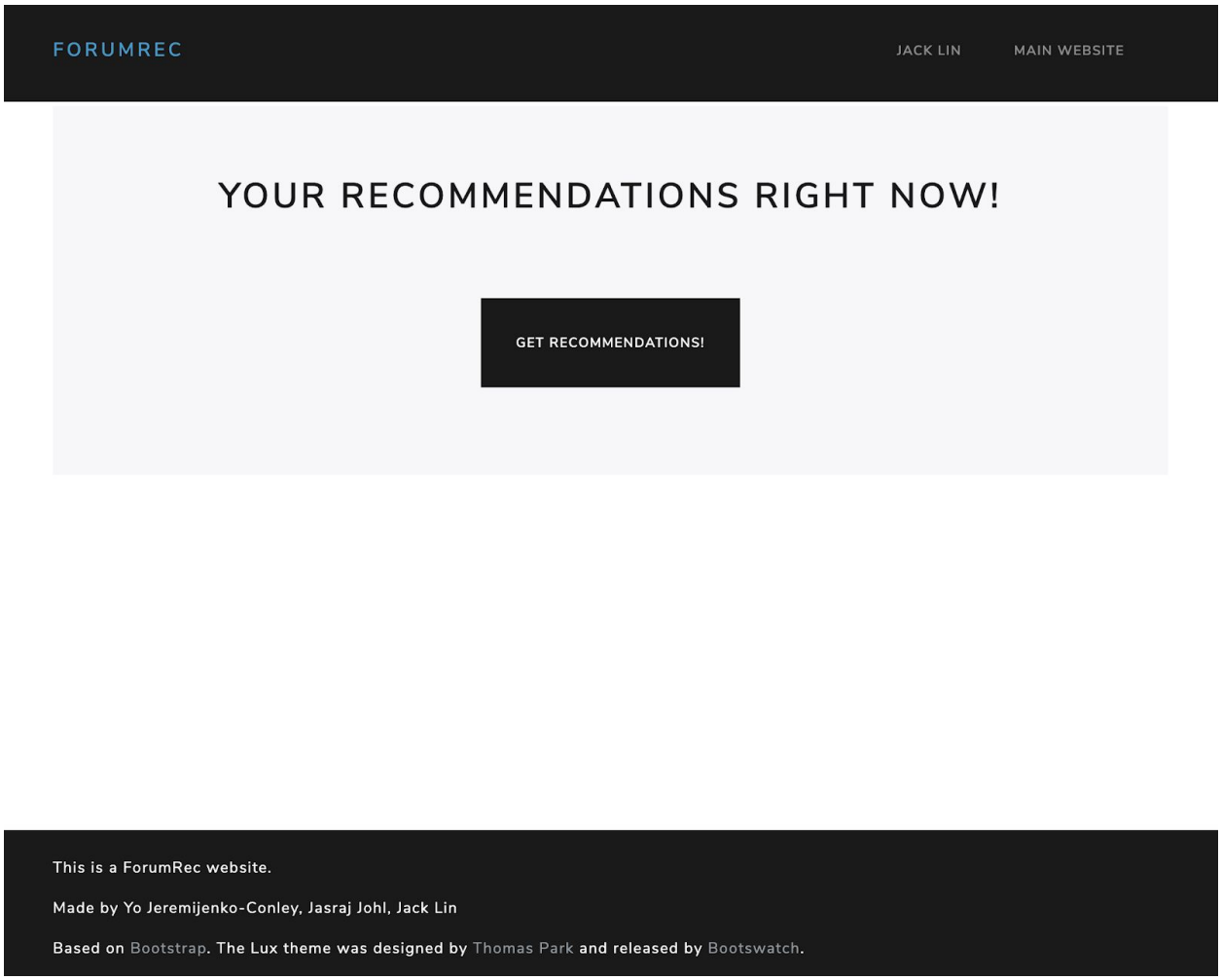
[8] Bootswatch. https://bootswatch.com/

[9] Lux Theme. https://bootswatch.com/lux

Appendix A1 - Screenshot: Landing Page



Appendix A2 - Screenshot: Getting Recommendations Once Logged In



Appendix A3 - Screenshot: Cold Start Question Selection