DS 410 - Midterm Project Report
Yuya Jeremy Ong & Yiyue Zou

Motivation

Many digital services today incorporate a certain form of recommender systems to provide the most relevant types of content to aid users to help discover new or similar items or content which may be relevant to their preferences. In the context of large scale data analytics, we find that building infrastructures and systems to support these types of operations are critical as retailers and content providers need ways to seek out preferences for a wide variety of options for users.

When building such recommender systems, online web services today are dawned with some of the following technical challenges and expectations from consumers. Large web services and retailers, such as Amazon, Youtube and Spotify, have huge amounts of data for both the type of items or content against a large number of users in their services [2]. This means, that the original representation of the data is quite sparse, as users are not likely to engage with more than 1% of the total entities of the product or content database [7]. Furthermore, balancing the discrepancies between providing a high throughput of recommendations and providing high quality recommendations are crucial to the user experience of the online services [2].

In our project, we will be focusing on these key challenges and perform a comparative analysis of the trade off between performance and quality of the recommendations. We will do this by systematically by breaking down each of the components of the different design options, hyper-parameters and feature data as a means of comparison. However, our emphasis in this project will be more focused on the optimization around the computational performance of constructing such models than evaluating accuracy of the recommendation (as recommendations are fundamentally qualitative). We will still however provide a qualitative demonstration of the provided recommendations between each of the models as a means of demonstrating the trade-off between each of the methods and the types of results they yield as a consequence of using certain features.

Problem Formulation

The objective of recommender systems is to provide a model to predict the initiation of a certain action for a given item/content for a particular user based on some features collected from the past pertaining to the users and the item/content. Depending on which modeling techniques we utilize, we may or may not exploit all of the provided features - which may significantly affect the overall runtime performance for the modeling process.

Formally, in a typical product recommender system, we have m users $U = \{u_1, u_2, u_3, ..., u_m\}$ and a list of n items, $I = \{i_1, i_2, i_3, ..., i_n\}$. Each user in U has a list of entities I_{u_i} for which we express the rating and/or sentiment about that specific entity. In the scope of recommendation, given a user $u_a \in U$, denoted as the active user, the task of the system is to provide the list of top N entities, $I_r \subset I$, in which the user will most likely favor, with the given constraint, $I_r \cap I_{u_a}$, that the user has not in any way engaged with that entity yet.

Design Options

For the recommendation algorithm, there are many different approaches to implement such methods to model recommendations. For our implementation, we will be focusing on building two different models which involve user-to-item and item-to-item relationships as our features. For each of the implementation options, we will be discussing briefly the modeling techniques used to implement each recommendation model, key features and parameters which are available for us to optimize around and some metrics which we will use to build comparative metrics within each models.

0.1 User-to-Item Recommender Systems

For the User-to-Item method, the relationship of the item and user is weighted by a combination of the product rating and the sentiment analysis of the product reviews. We evaluate the rating and sentiment as two distinct features, as there may be some discrepancy between the numerical rating and the sentiment of the reviews (i.e. with reliability of the reviews being another weighted component of the sentiment). The sentiment in this implementation will make use a existing corpus where we simply map the number of positive terms and negative terms to derive the frequency for each polarity and use those values to determine a score based on a similar method proposed by Gurini et. al [1]. This kind of "merit-based" ordering makes sense, since the higher the item is ranked the more pleasant the overall experience the user has with that given item, therefore increasing the probability that the user will be more likely to make that purchase. By aggregating such results across many users, we can build our predictive model based on the buying patterns of other user's and their overall reflection of the product themselves.

For our implementation of building a user to item based recommender system, we will use the Item-based Collaborative Filtering Algorithm proposed by Sarwar et. al [7]. Notably, this method evaluates the set of items the active user u_a has rated and/or reviewed and computes the similarity or correlation to the target item i and selects the top k most similar items $\{i_1, i_2, i_3, ..., i_k\}$, while concurrently, we also compute the similarity, $\{s_{i1}, s_{i2}, ..., s_{ik}\}$, for each of the corresponding objects. Thus the process of building this model is involved in a two step process, where: 1) we pre-compute the similarities between each items and 2) we generate the predicted outputs from the model. Pre-computing the similarities involve computing each items in the matrix by using some similarity metric such as cosine similarity, Pearson-r correlation and adjusted cosine similarity. Finally to generate our predictions, we can utilize two different techniques involving either a weighted sum or a regression based modeling technique. One method to derive such models can be through the use of methods such as SVD and regression based modeling techniques.

0.2 Item-to-Item Recommender Systems

In the Item-to-Item recommender system, we focus on the similarity of items, which in some respects is much harder to define. However, from our dataset, one way we can define the similarity of such objects based on the given semantic keywords provided by the product titles and description. In this implementation, we use a word embedding based model which allows us to learn these semantic features and provide us with a better feature representation of the product space [5] [6]. In this implementation we will build a model to first compute the feature vector of the product from the entire dataset. Then using the feature vector we have generated for our model, we can build an item-to-item based similarity matrix based on the computed features for each product. During our prediction phase, we can utilize this similarity matrix to generate our predictions based on using the past purchases of the user as a query to obtain the list of similar products. We can similarly generate these predictions based on similar methods proposed in the previous section given the similarity matrices.

Performance and Scalability

To evaluate the scalability of the presented models, we will be utilizing several different metrics to weigh out the discrepancies between the features and models utilized in the modeling process. Notably, both models require a two phase computational process where an offline pre-computing stage is performed prior to generating the predictions. For that, we will break up our time comparison analysis by measuring the pre-computing portion and the prediction separately during the evaluation phase. Within the notion of measuring prediction times, we will evaluate both the per query basis and the throughput of the model's performance as a metric for evaluating the runtime performance. However, within each of the given models, each modeling technique requires a different type of computation and therefore within each model, we will experiment with several different combination of features and hyperparameters to evaluate the performance in the overall runtime of the modeling.

In the User-to-Item model, to generate our model, we can select the k number of neighbors we define when finding similar users as we believe that this would factor most into the run time of the overall algorithm. During our prediction phase, we can comparatively analyze the differences in implementing the weighted sum and regression based models to see which one scales better and whether RDD based implementations of these methods are possible to further make use of the distributed environment.

In the Item-to-Item model, our modeling implementation relies on a word embedding feature model which relies on a iterative based method (notably performing SGD over the data). For this, we have the option to work with several different hyperparameters such as the number of dimensions for our feature vector, the window size and the learning rate which can greatly affect the runtime performance.

After constructing our baseline models across different combinations of hyperparameters and RDD based implementations, we will be further looking into specialized optimizations such as caching and data partitioning and job submit parameters to evaluate the general effect these changes have on the process. We will be systematically evaluating and organizing our observations to see which optimizations had the greatest effect to the computational runtime.

Milestones

For Lab 8, we will be focusing on the implementation of the User-to-Item based Collaborative Filtering algorithm. Our goal in the first lab is to build a working implementation of the proposed method as described above and run our experiments for different hyperparameters and configurations. In Lab 9, we will be working to implement our model for the Item-to-Item based model and again perform the same type of analysis done in Lab 8 for different hypreparameters and implementation options. Finally for Lab 10, we will be systematically experimenting with Spark's features by only focusing on non-algorithmic, but distributed optimizations on the entire modeling process. From this, we will take our existing code base from labs 8 and 9 to derive a more optimized method of generating results from changing external parameters independent from the algorithm itself.

Issues

One of the potential issues that we may foresee in our project is being able to take advantage of being able to take advantage of some of the optimization features which allow us to manage the underlying complexity of performing the computation under a distributed environment. Notably, we are not familiar with how to use some of the more advanced optimization features such as caching and data partitioning. During lectures and the labs, we are hoping to get some guidance and ideas on what are some of the best practices, concepts and ideas on improving our implementation from a distributed standpoint.

References

[1] D. Gurini, F. Gasparetti. 2013. *A Sentiment-Based Approach to Twitter User Recommendation*

[2] G. Linden, B. Smith, and J, York. 2013. *Amazon.com Recommendations - Item to Item Collaborative Filtering*

[3] J. McAuley, C. Targett, Q. Shi, and A. Hengel. 2015b. *Image-based recommendations on styles and substitutes.* In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR15). ACM, New York, NY, 4352. DOI: http://dx.doi.org/10.1145/2766462.2767755

[4] J. McAuley, R. Pandey, and J. Leskovec. 2015. *Inferring Networks of Substitutable and Complementary Products*

[5] T. Mikolov, *Distributed Representations of Words and Phrases and their Compositionality*

[6] Q. Le. 2014. *Distributed Representations of Sentences and Documents*

[7] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. 2001. *Item-based Collaborative Filtering Recommendation Algorithms*