QUAESTUS – A Top-N Recommender System with Ranking Matrix Factorization

Ajay Venkitaraman
K. J. Somaiya College of Engineering, Vidyavihar
Mumbai, India

Sahil Mankad
K. J. Somaiya College of Engineering, Vidyavihar
Mumbai, India

Umang Barbhaya
K. J. Somaiya College of Engineering, Vidyavihar
Mumbai, India

ABSTRACT
The last decade has seen rapid strides being taken in the field of recommender systems, which has been driven by both consumer demand for personalization as well as academic interest in implementing more accurate and optimized versions of recommender systems. In this paper we have discussed our implementation of Quaestus, a top-n item-based collaborative filtering recommender system with ranked matrix factorization (for relevance based sorting) which we have tested on an e-commerce dataset. We have used sentiment analysis to understand the polarity of reviews and thus extracting a score out of it, which in collaboration with the product rating (which was available on a scale of 1 to 5) has helped build a more robust recommender system. We have deployed Quaestus on an e-commerce website that we have built. The paper describes the phases of implementation and shows the method to deploy our model to the website that we have created. The results after experiments have shown that our model fares better than other algorithms with which we have compared our model.

General Terms
Recommender System, Natural Language Processing, E-Commerce Platform, Web Scraping, Machine Learning, NLP, ML.

Keywords
Recommender System, Matrix Factorization, Sentiment Analysis, Bigram Extraction, K-fold Cross-validation, Ranking Based Factorization.

1. INTRODUCTION
Recommender systems have gained prominence as an active area of research, especially due to the vast applications that it caters to. Researchers have been trying to build systems that can suggest products that precisely map consumers' requirements since the mid-1990s, with seminal research being undertaken on collaborative filtering [3], [4]. In its basic form, a recommender system tries to estimate ratings for products that a user has not seen before. Mathematically, it can be stated as follows: if C is the set of users and S is the set of all products (also referred to as items), and u is the utility function that measures the usefulness of item $s \in S$ to user $c \in C$, that is, $u : C \times S \rightarrow R$, where R is an ordered set. Here, we aim to choose the item that maximizes the user's utility, which when stated formally translates to the following [1],

$$s_c^{'} = \arg\max u(c, s) \ , \qquad where\ \forall c \in C, s^{'} \in S\ and\ s \in S$$

Primitive systems were replaced by better ones over the course of time, owing to both increase in computational capability and competition as technology startups used recommender systems as their 'unique selling point'. Today most of the business-to-consumer (B2C) firms implement their own recommender system to generate user-specific content, so as to serve their consumers better, and therefore are trying to one-up their competitors. In order to achieve this, research has focused on the algorithms and techniques to create recommender systems. Initial systems focused on content-based recommender systems, where the utility $u(c, s)$ of item s for user c is calculated based on the utilities $u(c, s_i)$ assigned by user c to items $s_i \in S$ that are 'similar' to item s [1]. This similarity is based on the 'content' of the product. For example, if we take food products, similar products can be food products of the same brand, having similar ingredients, etc. The inherent disadvantage of this method was that it was necessary to have a detailed metadata of the product so as to build the recommender system. Item-based collaborative filtering [6] was later used to overcome this disadvantage of content-based recommender systems. Here, the recommendations were based on item-item similarity measures (e.g. item-item correlation or cosine similarity between items). Relationship between items is identified for the computation of the most similar items, which is done using the set of items that the users have rated and/or reviewed. This was a key breakthrough in the age of e-commerce since purchase history of various users was available and the number of products whose information was available online proliferated.

The organization of this paper is as follows. Section 1 is the introduction to the paper. Section 2 describes the relevant information of the terminologies that are essential to understanding the paper. In section 3, we describe the system in its entirety and also its architecture. Section 4 is a discussion about the methodology and the key algorithms that we have used and is followed by the description of the datasets we used for our research in section 5. Section 6 showcases our results and section 7 concludes the paper. Section 8 is devoted to adding some of our ideas for future work in this space. In sections 9 and 10 we show our gratitude to the people who have made it possible for us to have implemented this research of ours.

2. LITERATURE SURVEY
Amazon [7] was one of the first major corporations to make use of item-based collaborative filtering, and other companies selling various products and providing various services soon followed suit. The Netflix Prize Competition conducted in 2006 by the company to improve their collaborative filtering showed that matrix factorization methods make the collaborative filtering based recommender systems more robust, accurate and help to make the process of learning easier in systems.

Matrix factorization methods essentially taps into the latent factors involved in analyzing a two-dimensional set (e.g. user-item set). Such latent factors sometime give us a lot more information with respect to the interaction between users and

the items they rate. Vectors of such latent factors are tested for correspondence, where a high value of correspondence would lead to a recommendation of the item to the particular user. These factors analyze both implicit and explicit user feedback. Implicit feedback refers to implicit user behavior, such as browsing patterns, time spent browsing a particular product, clickstream data, etc. Websites typically collect such information using personalization services and third-party HTTP requests [8]. Explicit user feedback is more readily available and is of a high quality, examples being ratings and reviews by users. Matrices formed using explicit user feedback is typically sparse, since every user can experience and comment upon only a limited number of products or services. The function of matrix factorization models is to map the users and the items into a latent factor space and model their interactions as inner products in the space [2].

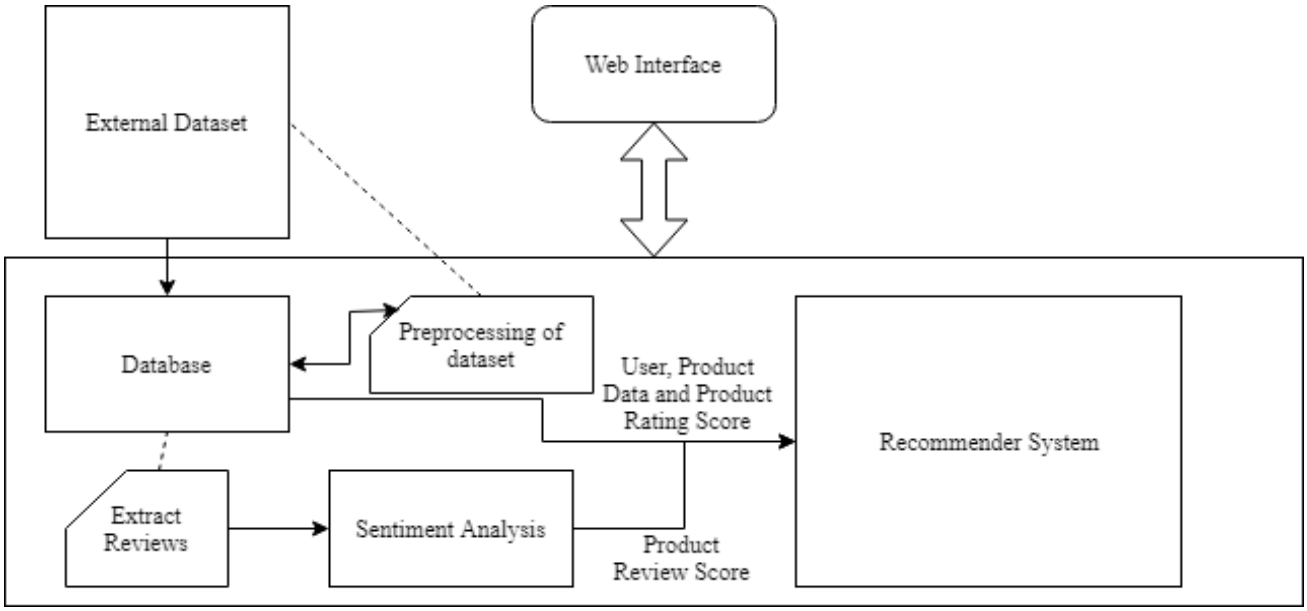

Fig 1: Architecture of the proposed system

The increasing trend of using machine learning methods in natural language processing, increasing availability of datasets of larger size and the betterment of processing power of devices lead to the rise of opinion mining and sentiment analysis since the last decade and a half. Sentiment analysis originally referred to the understanding of polarity in users' review. But in recent times, it has grown to encompass the computational treatment of opinion, sentiment used in the text and subjectivity and context of the same [9]. This application of sentiment analysis goes hand in hand with the requirements of recommender system, where it becomes a necessity to understand contextually what a user is trying to convey from a text which can be as short as a word to as long as a page or even more. Thus sentiment analysis has been employed a lot, of late in recommender systems and users are prompted to review products and services as much as possible.

3. ARCHITECTURE OF THE PROPOSED SYSTEM

The proposed system aims at developing the complete recommender model and creating an interface suitable for web browsing where users can log in and view products and find the recommendations generated for them. There are four phases of the architecture, viz. database creation phase, sentiment analysis phase, recommendation generation phase and user interface that displays the end results of the system. Each of these phases are explained in more detail along with the architectural representation (see Fig 1) below.

3.1 Database Creation Phase

In this phase, the database on which the subsequent phases will operate is created. This database can be created completely or modified by preprocessing existing open source datasets. Since creating complete datasets is a cumbersome task, modification of open source datasets from credible repositories is preferred for research purposes. The end result of this phase is a structured or unstructured database that can be used in the further steps (through different methods of querying the database). The key constituents of the database are user and product identification numbers, the rating given by the user to the product (represented in the form of a number) and the review provided by the user (represented as text).

3.2 Sentiment Analysis Phase

In this phase, sentiment analysis is performed on the review so as to understand its polarity and create a mapping of the emotion of the user behind every review. This falls under the umbrella of opinion-oriented information extraction [9], where the key takeaway is not to represent the product, but to represent what the user feels about the product. For our system, sentiment analysis being performed on the review makes the system more robust, since ratings don't fully capture the story. A 3-star rating can have a review that corresponds to a 3.5-star rating, while sometimes, it may correspond to a 2.5-star rating

This restriction imposed upon ratings due to the scale of the rating (5-star in our case) is offset by reviews since the mechanism of reviews gives the users a platform to convey their thoughts about the product. This step takes as input the review from the database and results in the generation of feature sets that help in detecting if the review is positive or negative (details of our implementation is specified in the following section).

3.3 Recommendation Generation Phase

This is the key phase of our system, where the recommendation of the 10 best products for every user takes place. The output of the previous phase (sentiment analysis

phase) is taken as the input of this phase, with a slight modification that the score generated for the recommender algorithm takes into account a combination of the review and the rating provided by the user. Here, we use top-n item-based collaborative filtering with matrix factorization enabled with relevance based sorting for developing more accurate predictions in an optimized fashion (n being 10 here). Moreover, this is a standard procedure used to compute recommendations from datasets that have a transaction list, since in such datasets, product information and the information about user-product interaction is more readily available, and thus item-based collaborative filtering and matrix factorization (to factorize the sparse matrices of ratings and reviews of products by users) blend well with such problems. The recommendations are ranked in descending order and a predefined number of recommendations (n) are provided to the user.

3.4 User Interface

A website has been created that acts as an e-commerce platform, where our recommender system has been deployed. The backend has been developed in Python 3 [10] since it provided us with rapid prototyping and the best available resources to optimize our implementation according to our needs. Specifically, we have used the Flask microframework [11], which is implemented in Python and is based on Werkzeug [12] and Jinja2 [13]. The user interface handles the functionalities in the website which includes but not limited to user login, signup, browsing of products, rating and reviewing them.

4. METHODOLOGY AND ALGORITHMS USED

This section talks about the specific implementation of the system along with a discussion about the algorithms used for the same.

4.1 Bigram Extraction and Preparation of Feature Set

In the sentiment analysis phase, we use a supervised learning approach [14]. In this approach, we first train our model on previously labeled reviews and then label new reviews based on the patterns extracted by the model from the training dataset. Several methods are specified in [14], from which we implement the sentiment analysis model using the Naïve Bayes Classifier. Firstly, we select n reviews each from the set of positive and negative reviews from an open source dataset and remove the punctuations (except the apostrophe, since it features in many words and removing it might change the meaning of the word). Then we stem the words to their root form using the Porter stemmer [16] so that all the variations of a particular word (with respect to plurality, tense, etc.) are transformed to be considered as the root word itself. For example: 'recommend', 'recommendation', 'recommends', 'recommended' and 'recommendations' are all converted to the root word 'recommend'. Bigrams from the context of natural language processing are consecutive units of words that are used in texts. According to [15], bigrams are particularly helpful in sentiment analysis as they have the ability to take into account modified verbs, nouns, etc. This makes it more logical to generate bigrams from the modified review text. As explained, it helps us classify modified forms of words (especially adjectives) such as 'not good' or 'not bad'. Not all bigrams are equally useful in providing information for sentiment analysis and it also becomes computationally expensive to use all the bigrams in the text. Thus we take into account the first 20000 most frequently occurring unique bigrams. The next step is to prepare feature sets since they help in discovering the frequently occurring bigrams in a review and also the label of the review. This helps our algorithm to decide if the bigram should be associated with a positive review or a negative one. For a previously unseen review, the algorithm checks whether the bigrams in it are more frequently associated with positive or negative reviews and labels the new review accordingly [17]. The feature set we use for our problem consists of a dictionary where the key represents the bigram and the value represents whether the bigram is a part of 20000 most frequently occurring unique bigrams.

4.2 Top-N Item-based Collaborative Filtering with Ranking Matrix Factorization

Top-N recommender systems are a modified form of recommender systems where the top n recommendations are provided (n is defined as 10 here). Item-based collaborative filtering aims at recommending items that are similar to the items that the user has rated previously. As stated above, this serves a better purpose than other collaborative filtering algorithms when product details are more readily available than user details (user-based collaborative filtering) and/or the constituents of the product (content-based collaborative filtering). Matrix factorization tries to make this process of finding similar items more accurate as well as more computationally efficient. It is a form of unsupervised learning since it aims at learning the latent factors in the relationship between users and items. This reduces the dimension of matrices to incorporate the most important factors, thus making it easier to compute the products to be recommended. We use a slightly modified version of matrix factorization, since we need to recommend multiple products to a user, and thus we need a metric to understand the best recommendations of the lot. Thus Quaestus uses a ranking method along with matrix factorization so as to sort the recommendations according to relevance and generate a score for every user-item pair and recommends items with the best score (in descending order). Let $\boldsymbol{i}$ represent a user and $\boldsymbol{j}$ represent an item, then the score can be given as:

$$score(i,j) = \square + w_i + w_j + a^T x_i + b^T y_j + u_i^T v_j,$$

Where $\square$ represents a global bias term, w_i represents the weight term for user i, w_j represents the weight term for item j, x_i and y_j represent the user and item side features respectively and a and b represent the weight vectors of the side features. The latent factors are represented by u_i and v_j. The minimization objective of the algorithm can be represented mathematically as:

$$\min_{w,a,b,V,U} \frac{1}{|D|} \sum_{(i,j,r_{ij})\in D} L\big(score(i,j), r_{ij}\big) + \lambda_1(\|w\|_2^2 + \|a\|_2^2 + \|b\|_2^2) + \lambda_2(\|U\|_2^2 + \|V\|_2^2) + \frac{\lambda_{rr}}{const * |u|} \sum_{(i,j)\in u} L(score(i,j), v_{ur}),$$

Where, D represents the observation dataset, r_{ij} represents the rating given by user i to item j, $U = (u_1, u_2, \dots)$ represents the user's latent factors and $V = (v_1, v_2, \dots)$ represents the item's latent factors, $L(x, y)$ represents the loss function and is given as $(x - y)^2$, λ_1 represents the linear regularization parameter and λ_2 represents the normal regularization parameter, λ_{rr}

represents the ranking regularization term necessary for ordering the recommendations, v_{ur} represents the rating value for unobserved items, and u represents the sample of pairs of unobserved users and items.

5. PREPARATION OF DATA AND TRAINING OF ALGORITHMS

This section talks about the datasets used for our research, the description of the same, the preprocessing of the data and the training of the algorithms mentioned in section 4 using the data.

5.1 Initial Product Dataset

As mentioned in 3.1, it becomes a tedious task to create full-fledged datasets from scratch for research purpose. To avoid this, our research uses the open source Amazon product dataset by Julian McAuley from University of California, San Diego [19].

5.1.1 Data Description

The McAuley dataset consists of over 160,000 rows of product details with 9 attributes, consisting of various product details, user rating, user review and identifiers of the row, like time of review. It was observed that some essential data like product name, product description (what the product actually is), image of the product, product price etc. was missing and this was necessary for the purpose of building our system in its entirety. Thus we had to perform Web Scraping from amazon.com using Python 3 [10] and some of its associated libraries such as lxml [20] and requests [21].

5.1.2 Data Preprocessing

The dataset so extracted in its raw form was not perfectly usable, since it contained noisy data from different encodings and it was necessary to clean the data so as to make it fit to be used as inputs to the various algorithms. Tuples with empty rows were removed, feature scaling was done, and attributes with high correlation were also removed so as to reduce redundancy and bias in predictions. The necessary features were incorporated in tables and the final product database was created.

5.2 Sentiment Analysis Data and Training the Model

The sentiment analysis module implemented in Quaestus uses a supervised learning approach. For this we needed a labeled dataset that contains reviews and their sentiments. We used the Amazon Reviews for Sentiment Analysis from Kaggle [22] for our purpose.

5.2.1 Data Description

The dataset consists of a few million Amazon customer reviews (input text) and their ratings in stars (labels) for training the sentiment analysis model.

5.2.2 Training the sentiment analysis model

The dataset was split randomly into train, test and validation sets in proportions of 70, 15, and 15 respectively. The validation set was used for hyperparameter optimization while the train set was used for training the model and test set was used to check the accuracy of the model. The data was shuffled before performing the split to avoid any skews that might emerge in the resulting split sets. The hyperparameter alpha (which is used for smoothing, to prevent any zero probabilities) was tuned using k-fold cross-validation (k taken as 5) by performing iterations with various values of alpha (as shown in Table 1) and the optimum value was used to train our model.

Table 1. Values of hyperparameter alpha and their corresponding accuracies

Sr. No	Alpha Value	Accuracy
1	0.01	73.587
2	0.1	75.027
3	0.5	76.373
4	1	77.0
5	5	77.507
6	6	77.613
7	7	77.587
8	7.5	77.587
9	7.75	77.573
10	7.875	77.627
11	8	77.64
12	8.125	77.64
13	8.25	77.64
14	8.375	77.6
15	8.5	77.613
16	8.75	77.587
17	9	77.587
18	10	77.547
19	15	77.253
20	20	76.947
21	25	76.707
22	50	76.013
23	75	75.733
24	100	75.573
25	150	75.053
26	500	73.427
27	1000	71.013

As evident from the table above, 8.25 is chosen as the optimum value of alpha to train the model. The test set accuracy of the model was around 81%. The model outputs a score of the review by deciding if it was positive or negative.

5.3 Training the Recommender System

The module that generates recommendations uses item-based collaborative filtering along with ranking matrix factorization for optimum accuracy and computational efficiency. The model outputs a predefined number of recommendations for a user and ranks them (sorts them) based on relevance. The module has been developed using Graphlab Create [18].

5.3.1 Data Description

The data for the recommender system is from the original database that gives the user and product identification attributes and the ratings given by the user for a particular product that is available in the dataset. The sentiment analysis module provides the score for every review. The review score and the rating is operated on to get the cumulative score that is used in the dataset that the recommender engine would use for its predictions.

5.3.2 Training the recommender system model

The dataset was shuffled randomly and then split into train and test sets using *k*-fold cross-validation (*k* taken as 5), so as to eliminate any bias in the dataset. The model was trained using Stochastic Gradient Descent (SGD) with some of the maneuvers suggested in [23] for improving the rate of convergence. The optimization is parallelized using multiple threads at once and this procedure being inherently random, the same parameters can give rise to slightly different models. So as to optimize hyperparameters, we use a random search space (set of parameters and their ranges), since choosing such random spaces can lead to better results within a fraction of computational time and budget as compared to manual search or grid search [24]. The results of those tests are used to finalize the values of the hyperparameters for training the model on the train set that is split by *k*-fold cross-validation. This trained model is used to generate the recommendations and the model is stored in the host machine.

6. RESULTS

The results were evaluated after plotting them on Jupyter Notebook [26]. The tables below (Table 2 and Table 3) show the precision and recall values respectively against the cutoff values (from 1 to 10) for the four models being compared, viz. Top-N Item-Based Collaborative Filtering with Ranking Factorization Recommender (hyperparameters optimized, denoted as M1), Ranking Factorization Recommender (with hyperparameters that have not been optimized, denoted as Model_1), Factorization Recommender (the conventional version, denoted as Model_2), Item-Based Collaborative Filtering (the conventional version, denoted as Model_3)

Table 2. Precision values of the algorithms at different values of Cutoff (up to 10)

Cut off	M1 Precision	Model_1 Precision	Model_2 Precision	Model_3 Precision
1	0.0205078	0.0263672	0.0332031	0.0195313
2	0.0224609	0.0234375	0.0253906	0.0205078
3	0.0221354	0.0205078	0.0224609	0.0195313
4	0.0231934	0.0214844	0.0224609	0.0205078
5	0.0230469	0.0210938	0.0216797	0.0214844
6	0.0236003	0.0216471	0.0214844	0.0221354
7	0.0241350	0.0217634	0.0216239	0.0230190
8	0.0244141	0.0209961	0.0211182	0.0231934
9	0.0246311	0.020833	0.0220269	0.0226780
10	0.025	0.0203125	0.0216797	0.0223633

Table 3. Recall values of the algorithms at different values of Cutoff (up to 10)

Cut off	M1 Recall	Model_1 Recall	Model_2 Recall	Model_3 Recall
1	0.0001271	0.0001701	0.0002103	0.0001286
2	0.0002806	0.0003016	0.0003227	0.0002667
3	0.0004198	0.0003936	0.0004261	0.0003790
4	0.0005885	0.0005471	0.0005660	0.0005300
5	0.0007356	0.0006699	0.0006839	0.0006971
6	0.0009023	0.0008261	0.0008136	0.0008613
7	0.0010790	0.0009698	0.0009561	0.0010364
8	0.0012511	0.0010755	0.0010648	0.0011891
9	0.0014197	0.0012011	0.0012540	0.0013073
10	0.0016031	0.0013027	0.0013733	0.0014344

As seen from the table, our model (M1) gives the best result in both precision and recall. M1 along with Model_3 are the only models where the precision increases with cut-off, thus showing the optimization of our matrix for top-n recommendations. The figures below (Fig 2, Fig 3, Fig 4, and Fig 5) show the precision and recall values of the algorithms across cutoff values (from 1 to 46) while performing *k*-fold cross-validation. They reiterate the fact that our model (m1) performs the best for our dataset. Some of the observations that we note here are as follows: The evaluation metrics (precision, recall, F1-score) are highly dependent on the dataset and any inaccuracies may also be caused by the dataset in question. Another important fact to be noted is that it is necessary to consider other top-n recommender systems so as to evaluate our results, and we have verified that they are comparable to similar algorithms [25], [27], [28] for top-n recommender systems. Our model (m1) has an increase in the precision as well as recall till a cutoff value of 10, but beyond the value of 10, the precision decreases as the recall increases, and even then, the precision as well as recall values are among the best in three out of the four folds in *k*-folds. Finally the most important observation to make here is that it is important to choose the correct evaluation metrics for recommender system (or more specifically, recommender systems that provide top-n recommendations) as metrics such as RMSE (root mean squared error) and MAE (mean absolute error) may not be the correct metrics, as they are generally regarded as regression metrics, i.e., metrics suitable to handle regression problems while the problem in our hand is a recommendation problem. Precision, recall and F1 score are still better than RMSE and MAE for such problems, but even they cannot be regarded as the best there is, as online evaluation metrics such as session abandonment rate, click-through rate, etc. are more popular for industrial use. Our model may not be the best in the above mentioned regression metrics, but it has proven to be so when it comes to precision and recall for our dataset.

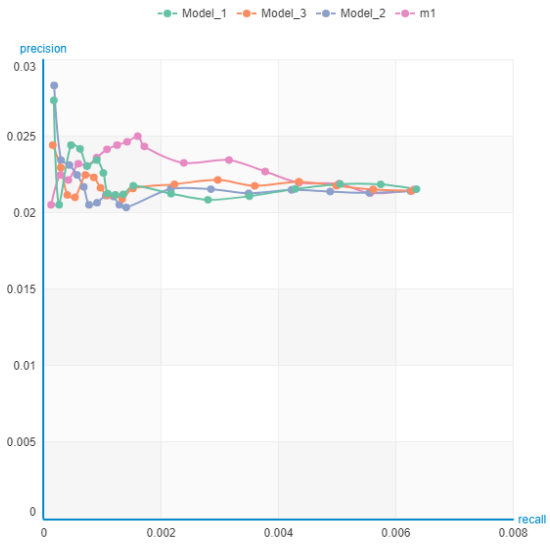

Fig 2: Precision recall values for different models across cutoff values (from 1 to 46) for fold 1

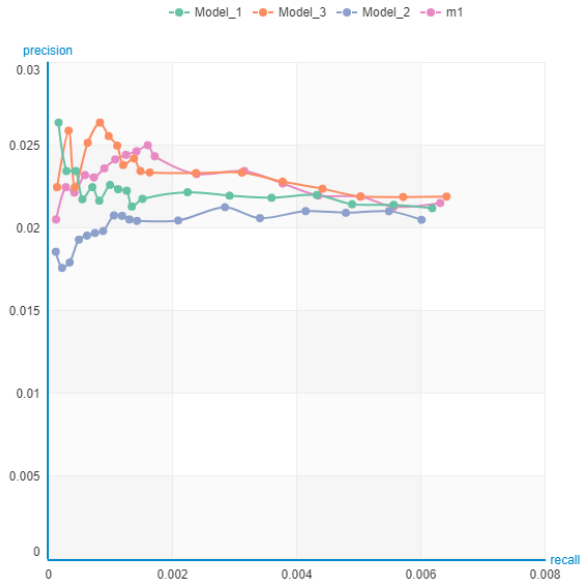

Fig 3: Precision recall values for different models across cutoff values (from 1 to 46) for fold 2

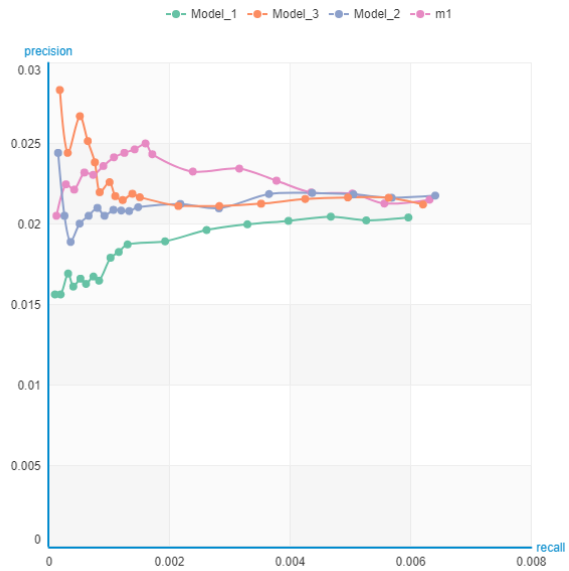

Fig 4: Precision recall values for different models across cutoff values (from 1 to 46) for fold 3

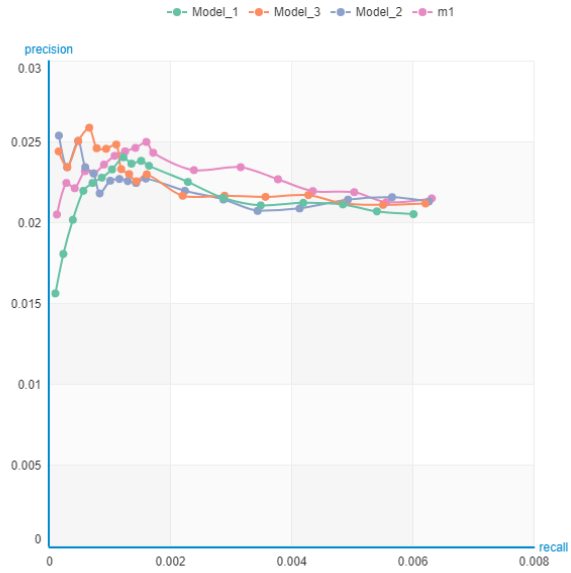

Fig 5: Precision recall values for different models across cutoff values (from 1 to 46) for fold 4

We can see here that the precision value for our model (m1) is around 0.025 and the recall value is about 0.0016 for a cutoff value of 10 and beyond a cutoff value of 10, the precision decreases gradually while recall increases.

7. CONCLUSION

The proposed system successfully recommends the top 10 recommendations for every user. The model that we proposed, viz. top-n item-based collaborative filtering recommender system with ranking matrix factorization has given us the best results for our dataset, and is comparable with other algorithms that have been proposed by academia [25], [27], and [28]. The optimizations that we have performed have

markedly improved the performance as compared to the model that was not optimized.

8. FUTURE SCOPE

Even though the performance of the model was on expected lines, it didn't fare very well in some metrics such as RMSE. This inaccuracy may be the result of the specific dataset and it remains to be seen if they can be overcome for other datasets. Also, a model that can recommend a variety of products across domains can also be thought of, which would necessitate the model to learn the various factors that lead to the purchase of a variety of products, since the consumer can buy products for very different reasons. Such an ensemble dataset may need rigorous train, test and hyperparameter optimization for performing competitively.

9. ACKNOWLEDGMENT

We are immensely thankful to our mentor Prof Rajni Pamnani, Assistant Professor at K. J. Somaiya College of Engineering for being supportive throughout the duration of our research and providing us with her immense expertise on the subject. We are also thankful to the researchers and academia across the globe, who with their research, have been a source of motivation and direction to our research.

10. REFERENCES

[1] B Adomavicius, Gediminas, and Alexander 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 (2005): 734-749.

[2] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 42.8 (2009).

[3] Konstan, Joseph A., et al. "GroupLens: applying collaborative filtering to Usenet news." Communications of the ACM 40.3 (1997): 77-87.

[4] Resnick, Paul, et al. "GroupLens: an open architecture for collaborative filtering of netnews." Proceedings of the 1994 ACM conference on Computer supported cooperative work. ACM, 1994.

[5] Girase, Sheetal, and Debajyoti Mukhopadhyay. "Role of Matrix Factorization Model in Collaborative Filtering Algorithm: A Survey." arXiv preprint arXiv:1503.07475 (2015).

[6] Sarwar, Badrul, et al. "Item-based collaborative filtering recommendation algorithms." Proceedings of the 10th international conference on World Wide Web. ACM, 2001.

[7] Linden, Greg, Brent Smith, and Jeremy York. "Amazon. com recommendations: Item-to-item collaborative filtering." IEEE Internet computing 7.1 (2003): 76-80.

[8] Puglisi, Silvia, David Rebollo-Monedero, and Jordi Forné. "On Web user tracking: How third-party http requests track users' browsing patterns for personalised advertising." Ad Hoc Networking Workshop (Med-Hoc-Net), 2016 Mediterranean. IEEE, 2016.

[9] Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis." Foundations and Trends® in Information Retrieval 2.1–2 (2008): 1-135.

[10] Van Rossum, Guido, and Fred L. Drake. The python language reference manual. Network Theory Ltd., 2011.

[11] Ronacher, Armin. "Welcome—flask (a python microframework)." URL: http://flask. pocoo. org/(visited on 02/02/2015) (2010): 38.

[12] Ronacher, A. "Werkzeug: The Python WSGI Utility Library." Release 0.9 (2011).

[13] Ronacher, Armin. "Jinja2 (the python template engine)." (2014).

[14] Neethu, M. S., and R. Rajasree. "Sentiment analysis in twitter using machine learning techniques." Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on. IEEE, 2013.

[15] Wang, Sida, and Christopher D. Manning. "Baselines and bigrams: Simple, good sentiment and topic classification." Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2. Association for Computational Linguistics, 2012.

[16] Porter, Martin F. "Snowball: A language for stemming algorithms." (2001).

[17] Bird, Steven, Ewan Klein, and Edward Loper. Natural language processing with Python: analyzing text with the natural language toolkit. "O'Reilly Media, Inc.", 2009.

[18] Dato-Team. (2013, October 15). GraphLab Create™. (C. Guestrin, Ed.) Seattle, Washington, USA: Dato, Inc.

[19] McAuley, Julian, et al. "Image-based recommendations on styles and substitutes." Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2015.

[20] Behnel, Stefan, Martijn Faassen, and Ian Bicking. "lxml: XML and HTML with Python." (2005).

[21] Reitz, Kenneth. "Requests: Http for humans." Online: http://docs. pythonrequests. org/.(24 December, 2012.) (2014).

[22] Bittlingmayer, Adam Mathias, Amazon Reviews for Sentiment Analysis, Kaggle Inc., Web, https://www.kaggle.com/bittlingmayer/amazonreviews/data (2017).

[23] Bottou, Léon. "Stochastic gradient descent tricks." Neural networks: Tricks of the trade. Springer, Berlin, Heidelberg, 2012. 421-436.

[24] Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." Journal of Machine Learning Research 13.Feb (2012): 281-305.

[25] Fazeli, Soude, et al. "User-centric Evaluation of Recommender Systems in Social Learning Platforms: Accuracy is Just the Tip of the Iceberg." IEEE Transactions on Learning Technologies (2017).

[26] Perez, Fernando, and Brian E. Granger. "Project Jupyter: Computational narratives as the engine of collaborative data science." Retrieved September 11 (2015): 207.

[27] Ye, Mao, et al. "Exploiting geographical influence for collaborative point-of-interest recommendation." Proceedings of the 34th international ACM SIGIR

conference on Research and development in Information Retrieval. ACM, 2011.

[28] Domingues, Marcos A., Alípio Mário Jorge, and Carlos Soares. "Using contextual information as virtual items on top-n recommender systems." arXiv preprint arXiv:1111.2948 (2011).

[29]