Proposal

John Bogacz
School of Engineering, University of Connecticut
Storrs, Connecticut
john.bogacz@uconn.edu

Nicholas Pang
School of Engineering, University of Connecticut
Storrs, Connecticut
nicholas.pang@uconn.edu

I. Introduction

Recommender systems have grown increasingly relevant as online services become more prevalent. The main objective of these systems is to be able to provide customers with content that they would enjoy. With a vast quantity of user-provided data and an even larger quantity of content to browse, it can be difficult for users to discover things that are new and interesting to them. Therefore, models serve as a means to help curate and reduce the information overload these services provide. While many different models and techniques have developed over the years, they typically fall under one of the following three categories: content-based filtering, collaborative filtering, and hybrid methods. Content-based filtering involves using an individual user's history of reviewing, browsing, or purchasing content in order to find similar items. Collaborative filtering involves using other users' history to find suitable content. Hybrid methods are a mix of the two.

Within collaborative filtering, one of the most popular methods is using matrix factorization to determine latent factors. Matrix factorization is built off of the concept of a user-item matrix, where users and items form the rows and columns of a matrix, respectively. By expressing this matrix as the product of a respective user matrix and item matrix, the model can learn a series of latent factors applied to all users and items at once. Many state-of-the-art models use this as a baseline embedding for users and items alike [2] [1] [5].

Matrix factorization takes advantage of the ratings that users give items in order to make predictions. Expansions have been made to solve some of the longstanding problems with this technique of collaborative filtering, such as sparsity. The sparsity problem arises due to the fact there are a large number of items that users will likely never rate. This makes it more difficult to find similar users to compare against. This problem is exacerbated for new users/users with very little reviews. One temporary solution commonly used is to remove users and items with sufficiently small numbers of corresponding reviews, allowing models to train on slightly less sparse data [3]. More commonly, models introduce more information into the system to draw more latent variables. Reviews have been a primary example of this, as they provide a rich source of text users provide justifying their rating. Models such as [2], [1], and [4] use various methods to condense the content of reviews and use those to influence the embedding of users and items. However, when examining the effectiveness of the models, they are often tested in a dataset that occupies a specific category. We do not agree with this method of evaluation, as it captures a minimized view of user behavior rather than a holistic one, and therefore may miss some overarching patterns otherwise hidden.

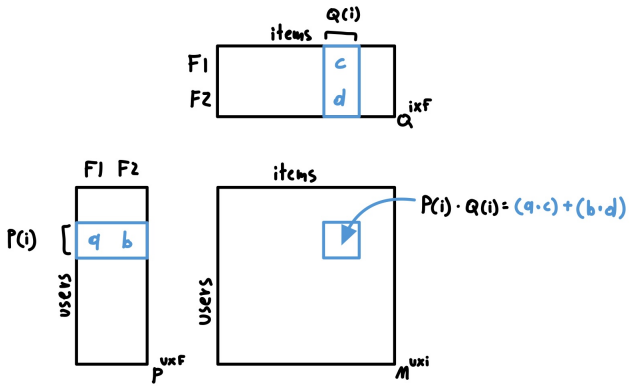

Fig. 1. Matrix factorization.

Another factor often overlooked in collaborative filtering methods and offer additional information to better define user preferences are item descriptions. Item descriptions can help find hidden patterns within user behavior that are not expressed within user reviews. For example, if a user likes an item that falls under a particular brand, they might not write the name of the brand in their review. However, by examining the item descriptions, it is possible to notice this pattern.

In our research, we plan on exploring the following:

1) We would like to measure the performance effects of various topic-generating algorithms on item recommendation.
2) We want to observe how these algorithms perform on more generalized item databases.
3) We would like to evaluate the effectiveness of these techniques on the introduction of new users into the system.

II. Related Works

A. Matrix Factorization

Many different techniques to utilize collaborative filtering have been developed within the recommender system subfield. Most models that use collaborative filtering take advantage of matrix factorization in order to reduce the complexity of the

model and learn latent factors [1], [2], [5]. As seen in Fig. 1, matrix factorization works by treating the user-item matrix $R \in \mathbb{R}^{u \times i}$ as a product between users $P \in \mathbb{R}^{u \times k}$ and items $Q \in \mathbb{R}^{i \times k}$. K, in this case, is the number of latent factors the model will consider, defined by the user.

One potential issue with this is that the data is not normalized, since ratings are from numbers 1 to 5. To prevent this causing significant error, most implementations of matrix factorization introduce biases. The biases serve as means of allowing the matrix factorization to predict normalized data, reducing error. The updated calculation is:

$$R(u,i) = p_u \cdot q_i^T + u_b + i_b + g_b$$

- p_u is the embedding of user u from P.
- q_i is the embedding of item i from Q.
- u_b and i_b are user and item biases.
- g_b is the global bias.

B. Latent Semantic Analysis

Latent Semantic Analysis (LSA) [7] is an advanced technique in order to abstract out the hidden context or topics within a document. The benefits of using LSA is it can reduce the dimensionality of the original text corpus, analyze word associations in the text corpus, and find relations between terms. The LSA algorithm assumes that words that are closer in connotation will then occur in similar documents. Within the LSA method, we start with a matrix that's a width of the number of documents while the height is the number of unique words used in all the documents. There are different methods for filling out each element in the matrix, since each element in the matrix represents a particular word in a particle document you can represent this relation as the frequency of the word, binary representation if the word exists, or use log entropy. At the end of this process we want a scalar quantity representing the similarity between the all document vectors over all the terms in the text corpus. The algorithm performs a singular matrix decomposition (SMD). SMD then represents a matrix as the following.

$$XX^T = U\Sigma V$$

- U is a matrix that represents how the rows or documents are related to each other.
- Σ is an identity matrix where the values where the number one is replaced with different constants across the diagonal and each value represents the strength of the relationship between a column in U to a row in V.
- V is a matrix that represents how the columns or words are related to each other.

After all this work we can finally perform singular value decomposition (SVD) on X for each document and get a vector for all documents. Some of the drawbacks of LSA is that we need a lot of words in our documents in order to get a better representation of similarities between different documents as well.

C. Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TF-IDF) [8] is similar to the Latent Semantic Analysis (LSA) method mentioned previously. It creates a matrix that's the width of the number of documents and height is the number of unique words used in all the documents. TF-IDF then assigns a weight for each word in each document as the following.

$$t \cdot f_{i,j} \cdot \log\left(\frac{N}{df_{i,j}}\right)$$

- t is the weighting factor that can be used to adjust the final weighted score.
- $f_{i,j}$ is the number of occurrences of a term in a specific document.
- N is the number of total documents.
- $df_{i,j}$ is the number of documents containing that specific word.

This tweak to the existing LSA method is very useful for the fact that TF-IDA can enhance document representations by ensuring items are weighed more heavily, as well as weighing down less informative terms thereby reducing noise.

D. Probabilistic Latent Semantic Analysis

The Probabilistic Latent Semantic Analysis (PLSA) [9] method differs as it uses a probabilistic model with latent topics to generate the data observed in the document-term matrix. In order to accomplish this the algorithm needs to find the probability P(d,w). The algorithm relies on the assumption that each document consists of a variety of different topics and each topic relies on a set of words. The algorithm needs to calculate P(Z—D) which is the probability of topic z is present in a given d document, P(W—Z) the probability that a given word is present in a topic z. Both of these probabilities are modeled as multinomial distributions are trained on expectations-maximization (EM) algorithm, where EM is a way of finding a likelist parameter estimate for a model on unobserved latent variables or topics. Finally to calculate P(d,w) we set it equal to the join probability of seeing a given document and word together which is becomes.

$$P(D)\sum_Z P(Z|D)P(W|Z)$$

The equation lets us know how likely it is to see a document and how likely it is to find a certain word in that document. Interestingly enough, another way to write this equation is in the form of...

$$P(d,w) = \sum_Z P(Z)P(D|Z)P(W|Z)$$

In this new form, the similarities between PLSA and LSA are shown below.

- $P(D|Z) = U$
- $P(Z) = \Sigma$
- $P(W|Z) = V^T$

Some downsides to PLSA is that the model is prone to over fitting.

E. Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [10] is a probabilistic generative model used in NLP and ML for topic modeling, and is designed to discover underlying topics in a collection of documents and understand how words are distributed across these topics. LDA is a bayesian version of PLSA. The DLA uses the dirichlet distribution which is a distribution over distributions. LDA relies on the assumption that each document is a mixture of various topics, and each topic is a distribution of words. LDA assigns words to topics probabilistically, and documents to topics based on the occurrence of these topics in the document. The main idea is that the LDA model infers the latent topics and their word distributions that are most likely to have generated the observed documents in the corpus. LDA is preferred over PLSA as it can generalize to new document easily, while in PLSA document probability is fixed the LDA model the dataset serves as a training data for the dirichlet distribution of topic-document distributions.

F. Other Applications of Textual Data - Explanability

Beyond predicting user ratings of products, recent models have also made attempts at providing explanations as to why items are recommended. By providing explanation text, the model can help users build trust in the system's decision making, aiding the effectiveness of the model. This subset of recommendation systems is called explainable recommendation. [1] includes a layer that generates the relative importance of reviews in its prediction for the rating. These numbers can be extracted to find the most representative reviews the model takes in in making its prediction, essentially serving as the explanation. Worthy of note is that it is limited in the kinds of explanations it can provide, as it only provides existing review text and does not paraphrase. [4] uses the reinforcement learning approach to generate more complex explanations that match a set of predefined rules. Rather than predicting user ratings, it learns the predictions of another model within the recommender systems topic, with agents matching their output so that they match the output of that model. The reinforcement learning framework allows it to not only explain the way in which the underlying model itself works, but also modify the explanation text. Both [1] and [4] use CNNs to embed reviews based on the textual content.

III. Proposed Directions

At the present, we have extensively tested matrix factorization with various optimizations. We found that the best improvements was through regularization of the parameters, which drastically improved the convergence speed of testing accuracy, as seen in Fig. 2 and Fig. 3. Other changes, such as introducing dropout, adjusting the learning rate, and having more latent factors than 5 appear to have a significant impact on testing accuracy in the long term.

We propose a research plan in which we aim to explore various topic generation techniques for efficiently incorporating textual information into a recommendation system in order to increase prediction accuracy. Our goal will be to test at most 3

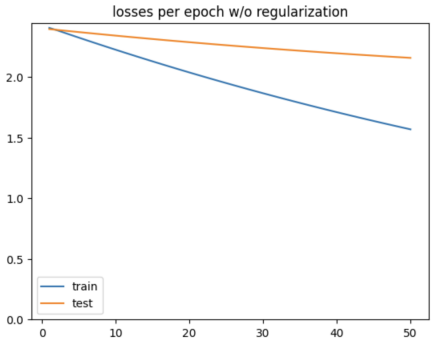

Fig. 2. Matrix factorization performance without regularization.

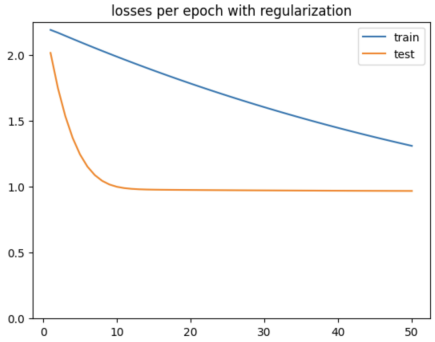

Fig. 3. Matrix factorization performance with regualrization

different models mentioned within our related works sections in order to gain a deeper insights on hidden user-generated texts from reviews and item metadata, to increase prediction accuracy and enhance overall recommendation quality.

Previous work such as [2] explore topic genre discovery, but only impose these onto the overarching categories defined by Amazon. This is an unrealistic representation of user buying patterns, as a user's decision making can apply across these categories and influence products from other genres entirely. We hope that introducing more varied data would allow for a more generalized and robust solution.

Additionally, we would like to directly use topics to aid in the rating prediction process. Papers such as [3] have pointed out that review text is best used as a regularization tool like in [2] rather than additional information in works such as [1]. However, the techniques they mention that use review text to predict mainly create embeddings over the entire review text. We hope that by using topic distributions in the prediction process, we can better integrate review text and other relevant text into the prediction process.

We will first explore existing topic analysis solutions. To this end, Latent Dirichlet Allocation (LDA) [10] has shown to be very promising as it utilizes more rigorous and mathematically principles over other forms of measuring document similarities approaches like Latent Semantic Analysis (LSA) [7] as well as positioning itself as a more advanced iteration of Probabilistic Latent Semantic Analysis (PLSA) [9]. Because LDA utilizes

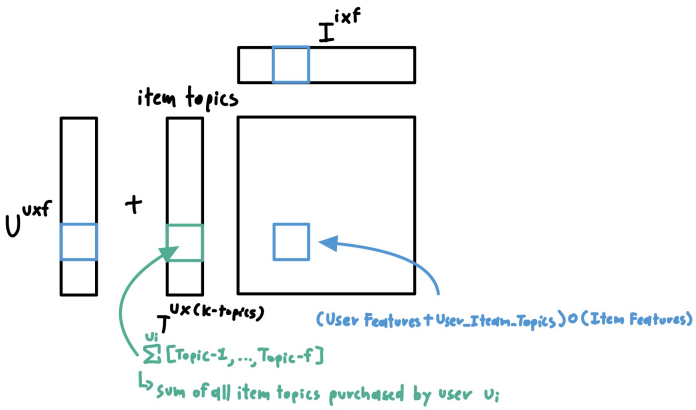

Fig. 4. An example of using LDA within a matrix factorization framework.

TABLE I
SIZES OF PROPOSED DATASETS

Category	#Users	#Items	#Reviews
Video Games	24303	10672	231780
Android Applications	87271	13209	752937
Health and Personal Care	38609	18534	346355

a Bayesian framework, it can leverage the Dirichlet distributions to allow it to uncover latent topics through their word distributions. We plan on utilizing this processes within our model as a framework upon which we can fine tune smaller details in order to increase model prediction accuracy. This aspect of LDA is instrumental in understanding the underlying structure of documents and how words are distributed across the documents. In summary, the research presented in [10] demonstrates a highly promising technique in NLP, and we will be incorporating it into our matrix factoring model and building on top of it.

Fig. 4 presents a simple example representation of how we could incorporate topic distribution models into matrix factorization. In this case, we use the topic distribution of a user's past items to transform the user embedding vector. In doing so, we have a better representation of a user that when trained alongside the item embedding can improve the overall result. We can also do this for the item embedding as well.

An important part of judging the performance of these models is being able to have a good understanding of how the model categorizes items. Throughout the research we will frequently analyze how changes affect the overall categorization. LDA makes this simple to carry out, as it uses words within the review text to estimate probabilities of topics, meaning that extracting those words give a fundamental understanding of the categories it created.

IV. DATASETS

Datasets we will consider for this project are from Amazon's 2014 review dataset, which is a publicly available dataset. Because we would like to examine the performance of topic modeling across more varied categories, we will select several from this dataset. However, in the interest of memory constraints, we will not select all of them. To model similar and vastly different categories, we choose Video Games, Android Applications, and Health/Personal Care. Video Games and Android Applications are considered similar categories, and Health/Personal Care is considered vastly different than these two. The Video Games category contains over 230,000 reviews between 24,000 users and 10,000 items. The Android Applications category contains over 750,000 reviews between 87,000 users and 13,000 items. The Health/Personal Care category contains over 340,000 reviews between 38,000 users and 18,000 items. See Table I for more specific numbers.

Each of these categories come with the following metadata that we find relevant:

1) Reviews: this contains all of the reviews made between users on items within the specific category. Relevant variables within these are the user ID, item ID, review text content, and rating, expressed as an integer from 1 to 5.
2) Item descriptions: this contains the descriptions for all of the items within the specific category. Relevant variables within these are item ID, item description, item title, and predefined categories. These categories are an array of words.

The Amazon datasets have been preprocessed to remove any duplicate items, as well as removing any users and items with less than 5 reviews associated with them. Alternatively, the public dataset also contains the raw datasets for each category, where items and users can have as little as one item or review associated with them. These datasets are a better representation of realistic scenarios, but take up significant space that our current resources will not be able to operate on in a preferable time. However, we will keep the existence of these datasets known in the event we choose to test the impact of sparsity on topic generation.

We will impose our own preprocessing on these datasets in order to conform to some of the models that we will be using. Since many of our components have an embedding feature, we map our user and item IDs to ordered lists starting from 0. This makes it easier to perform embedding retrieval and keeps the size of the arrays low. Additionally, we calculate the global bias of all reviews beforehand before perform training.

V. RESPONSIBLE CONTENTS

Within the proposal paper, Nicholas worked on the introduction and datasets, John worked on the related work, and the proposed directions were discussed and completed as a group. We plan on working together and having weekly meet-ups to discuss current project successes as well as ongoing plans to complete the project in a timely manner.

REFERENCES

[1] C. Chen, M. Zhang, Y. Liu, and S. Ma, "Neural Attentional Rating Regression with Review-level Explanations," Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18. ACM Press, 2018.

[2] J. McAuley and J. Leskovec, “Hidden factors and hidden topics,” Proceedings of the 7th ACM conference on Recommender systems. ACM, Oct. 12, 2013.
[3] D. Roy and M. Dutta, “A systematic review and research perspective on recommender systems,” Journal of Big Data, vol. 9, no. 1. Springer Science and Business Media LLC, May 03, 2022.
[4] X. Wang, Y. Chen, J. Yang, L. Wu, Z. Wu, and X. Xie, “A Reinforcement Learning Framework for Explainable Recommendation,” 2018 IEEE International Conference on Data Mining (ICDM). IEEE, Nov. 2018.
[5] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and M. Zhou, “A Deep Latent Factor Model for High-Dimensional and Sparse Matrices in Recommender Systems,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 7. Institute of Electrical and Electronics Engineers (IEEE), pp. 4285–4296, Jul. 2021.
[6] L. Zheng, V. Noroozi, and P. S. Yu, “Joint Deep Modeling of Users and Items Using Reviews for Recommendation.” arXiv, 2017.
[7] T. K. Landauer, P. W. Foltz, and D. Laham, “An introduction to latent semantic analysis,” Discourse Processes, vol. 25, no. 2–3. Informa UK Limited, pp. 259–284, Jan. 1998.
[8] H. Christian, M. P. Agus, and D. Suhartono, “Single Document Automatic Text Summarization using Term Frequency-Inverse Document Frequency (TF-IDF),” ComTech: Computer, Mathematics and Engineering Applications, vol. 7, no. 4. Universitas Bina Nusantara, p. 285, Dec. 31, 2016.
[9] T. Hofmann, “Probabilistic Latent Semantic Analysis.” arXiv, 2013.
[10] R. Parizotto, B. L. Coelho, D. C. Nunes, I. Haque, and A. Schaeffer-Filho, “Offloading Machine Learning to Programmable Data Planes: A Systematic Survey,” ACM Computing Surveys, vol. 56, no. 1. Association for Computing Machinery (ACM), pp. 1–34, Aug. 26, 2023.