CSE 5819 Final Report - Sentiment Analysis in Recommendation Systems

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.

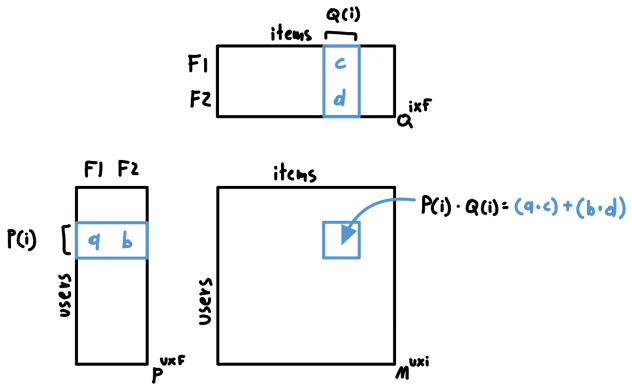

Fig. 1. Matrix factorization.

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 present the following:

1) An analysis on various inclusions of LDA within a MF-based model.
2) A comparison of different scopes of databases to understand categorization better.

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 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 document set, analyze word associations in the text document set, and find relations between terms. LSA assumes that words that are closer in meaning 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 and this is what is used to measure the similarity of documents. But, 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.

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

- t is the weighting factor that can be used to adjust the final 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 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 assumes 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 and are trained on expectations-maximization (EM) algorithm. EM is a way of finding a likeliest 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. One other 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.

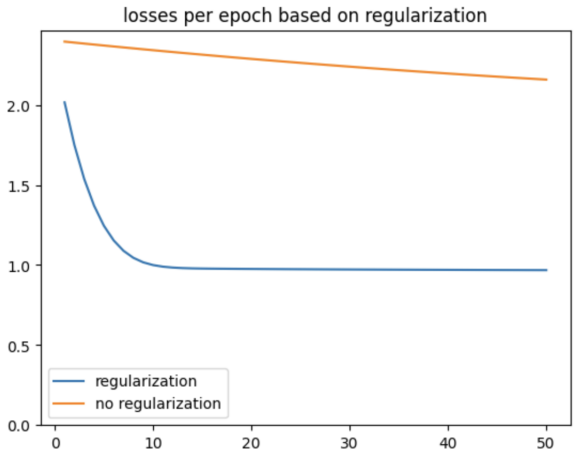

Fig. 2. Matrix factorization based on weight regularization.

E. Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [10] is a probabilistic generative model for topic modeling, and is designed to discover topics in a collection of documents and understand how words are distributed across those topics. LDA is a bayesian version of PLSA and uses the dirichlet distribution, which is a distribution over distributions. LDA assumes that each document is a mixture of various topics, and each topic is a distribution of words. LDA assigns words to topics probabilities, and documents to topics based on the frequency of these topics in the document. The main idea is that the LDA is that the model infers the latent topics and their word distributions that are most likely to have generated the observed documents in the document set. LDA is preferred over PLSA as it can generalize to new document easily, while in PLSA document probability is fixed.

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

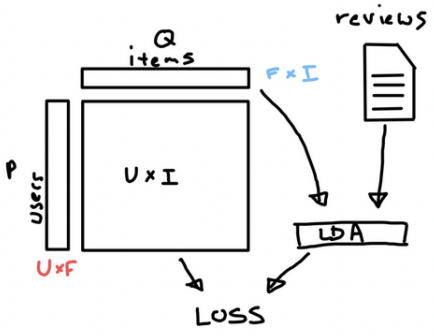

Fig. 3. LDA as a regularizer.

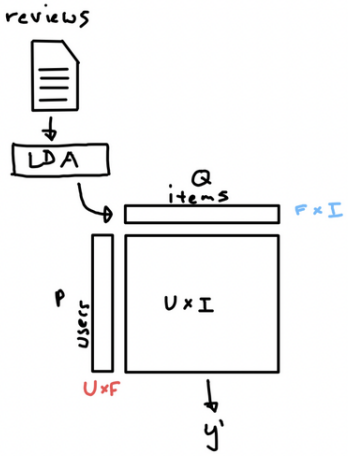

Fig. 4. LDA as an initializer.

explanation text. Both [1] and [4] use CNNs to embed reviews based on the textual content.

III. Technical Components

Our work involves incorporating Latent Dirichlet Allocation into our model using various methods. The topic distributions per document that LDA generates can be extrapolated to the latent factors generated by matrix factorization. LDA's inverse document frequencies has a dimensionality of R^{DxT}, where D represents the number of documents and T represents the number of topics, while MF's item embeddings have a dimensionality of R^{IxF}, where I represents the total number of items and F represents the latent factors. Each item can be considered a document, with the concatenated reviews being considered the text within the document, while the topics can be synonymous to latent factors. This makes translating between LDA and MF trivial.

Fig. 1 represents our base case, matrix factorization. We have extensively tested this with various optimizations to understand better how to optimize its performance. We found that the best improvement was through regularization, which drastically improved convergence speed, as seen in Fig. 2. Other changes, such as introducing dropout, adjusting the

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

learning rate, and having more latent factors than five, do not appear to impact overall model performance.

Following this, we apply two different changes involving LDA into the model. First, as outlined in Fig. 3, we use LDA to calculate a parameter that acts as a regularization variable within the loss function. This directly mirrors the work done in [2] and the intuition we generated on our own. The regularization parameter is drawn from the log-likelihood of the text corpus. We can calculate the log-likelihood of the embeddings given the existing review text per item using the corpus likelihood function:

$$p(C|\Theta,\Phi) = \prod_{d}^{T}\prod_{w}^{N}\Theta_d\Phi_w$$

In this case, T is the total number of documents, synonymous with the number of items, N is the number of words per topic, Θ is the topic distribution for a particular document, and Φ is the word distribution for a topic corresponding to the document.

The conversion from corpus likelihood to perplexity is as follows:

Secondly, we use the topics generated by LDA to initialize the weights within the model directly. 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 for direct model calculations do not involve topic mining but rather text analysis using NLP methods such as CNN and attention. We would like to re-explore this category with LDA directly contributing to the model predictions using the topics generated as initial weights.

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

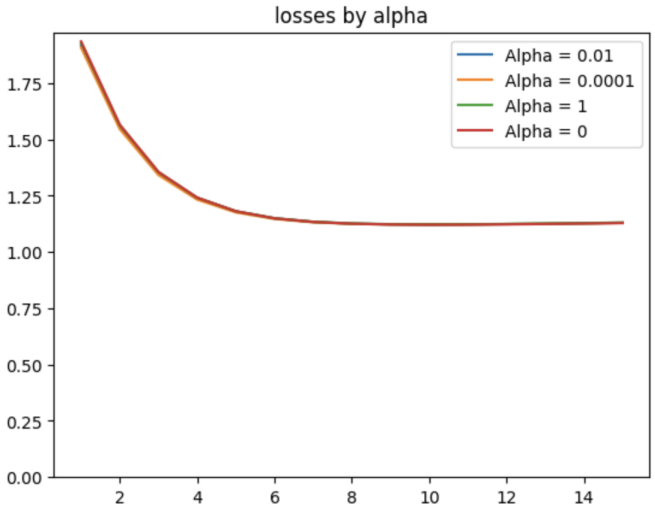

Fig. 5. Loss Based on LDA Alpha Value.

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,

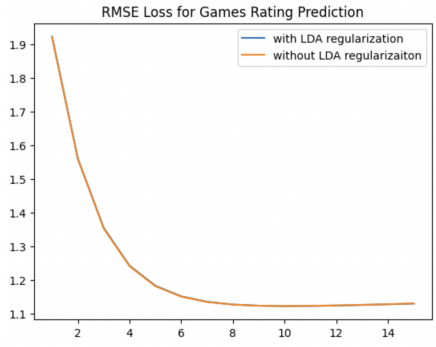

Fig. 6. LDA Regularization on the Amazon Toys and Games Dataset.

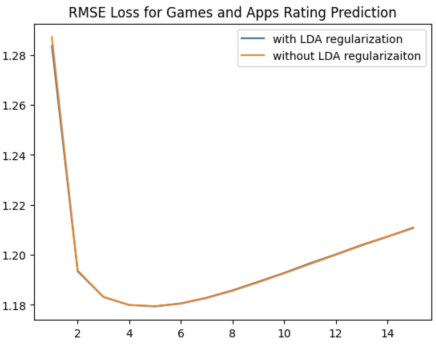

Fig. 7. LDA Regularization on the Amazon Games and Android Applications Datasets.

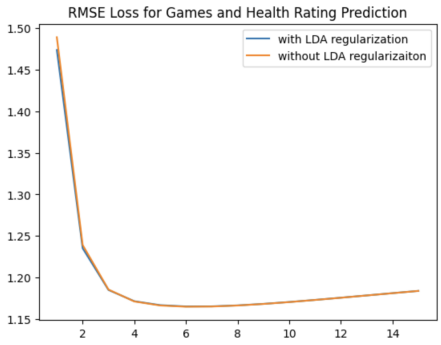

Fig. 8. LDA Regularization on the Amazon Games and Health Datasets.

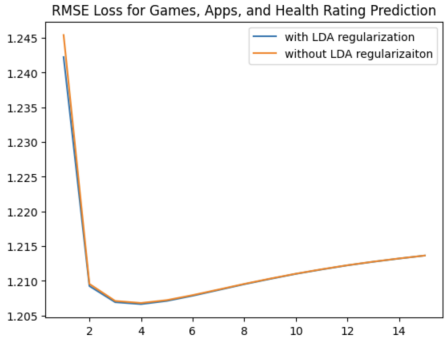

Fig. 9. LDA Regularization on All Datasets.

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. Experimental Settings

Our project was conducted using MacOS on M1 and M2 chips. We coded primarily in Python using the Skorch and PyTorch libraries. One of the biggest limiting factors throughout the project was the speed of the model. LDA took a significant amount to run, incrementing the speeds of each epoch by a factor of 100. Because of this, we were unable to properly cross-validate the model parameters and stuck with a simpler model. In order to run our experiments in a reasonable amount of time, we used a batch size of 8192, 5 total factors for both MF and LDA, a learning rate of 0.01, 15 epochs, weight decay of 0.001, dropout of 0.4, and a weight factor of 0.001 for the LDA regularization parameter. Because many of these factors seem to exhibit no noticeable effects on the performance of the model, we decided to change everything except for the dropout to make the calculations run faster.

For the LDA, we excluded stop words from the vocabulary, while keeping in all other strings. That way, we can avoid including words such as 'to' and 'from', while included more niche words such as 'Playstation' and '500GB', which are not actual words in the English dictionary but also contain significance in a buyer's decision making. Unfortunately, this also meant that it was impossible to rule out misspellings and other grammatical errors. This proved to be a problem in our final results.

VI. Results

A. LDA as a Regularization Parameter

Because of the additional term in the loss calculation, the training loss is heavily impacted by the introduction of an additional regularization parameter. This can clearly be seen in Fig. 6-Fig. 12, as the lines are overlapping. However, in practice, the introduction of LDA regularization does not appear to impact the overall performance of the model, as displayed in Fig. **??**. One potential explanation is that because the LDA perplexity calculation is based already off of the matrix factorization weights, the similarity is enough such that the regularization parameter is only amplifying loss. Another potential explanation is that the hyperparameters are not optimized, but this is not something that can be explored for this project.

Interesting behavior is observed between different scopes of datasets, as seen in Fig. **??**. As the dataset broadens in category, the convergent loss is higher. This is expected, as intuitively, a larger vocabulary would require more categories, of which we cannot provide due to processing speed constraints.

B. LDA as Model Weight Initialization

One of our proposed directions we intended to utilize the LDA to initialize the values of the user and item embedding's

layer. The way this was done was to train the LDA on a corpus of text and to extract the user and item topics distribution. Afterwards we mapped our user-id's and item-id's from the raw data-set containing hashed strings from Amazon to integers which would prove to be useful when initializing the weights for each row in the embedding's layer. The final adjustment before running our hypothesis model against the baseline is to create another mapping by adjusting the topic distribution of the users and items. Since by default Torch's embedding's layer initialized the weights randomly from a range from -1 to 1, we mapped the topic distribution for each item and user such that the largest value is equal to 1, smallest value is equal to -1, and all values in between are mapped accordingly.

After running the LDA on reviews based on the user's account we see a glimpse into the online behavior of internet users on the Amazon platform.

- **Topic 0: Games**
  - **game**, **app**, **fun**, **play**, love, like, **kindle**, games, great, really, time, just, good, free, **playing**, don, easy, **graphics**, recommend, enjoy, awesome, use, think, challenging, way, lot, **played**, **levels**, want, quot.
- **Topic 1: Product Reviews**
  - **product**, use, **good**, **great**, **like**, **used**, just, **using**, ve, **works**, **price**, **work**, does, really, don, 34, **better**, day, taking, **feel**, time, **taste**, **razor**, years, clean, **bought**, water, little, tried, did.
- **Topic 3: Toys**
  - **toy**, old, **loves**, **years**, **son**, **daughter**, **kids**, **little**, set, **christmas**, great, bought, **toys**, **play**, **doll**, **grandson**, pieces,**gift**, cute, **fun**, **child**, 34, love, got, **loved**, just, **birthday**, **granddaughter**, really, **quality**.

Additionally we ran another LDA on review based on each item and this has given us in insight on the groupings of what reviews are like based on each item.

- **Topic 0: Gaming Apps**
  - **game**, **app**, **fun**, **play**, **games**, kindle, like, love, free, just, time, really, **playing**, good, great, **graphics**, don, **puzzles**, **played**, **puzzle**, quot, **apps**, **easy**, **challenging**, version, screen, enjoy, **levels**, cards, **download**.
- **Topic 1: Toys**
  - **product**, use, like, great, just, **toy**, good, **little**, **old**, really, year, 34, time, loves, used, set, price, don, bought, love, work, works, **son**, ve, day, using, **daughter**, does, **kids**, better.
- **Topic 4: Health and Wellness**
  - potholders, **tecnu**, **omegavia**, **omapure**, **mgsdha**, **mgsepa**, **mgspercentage**, urinals, mgstotal, **innovix**, formnot, slingo, 1other, **vegetarianifos**, clarinet, **mercurymolecularly**, **ifos**, **nclex**, **chaga**, minami, potholder, liana, moen, **gmoscontains**, babbel, distillednot, ourworld, **hydraplenish**, rosalie, twistables.

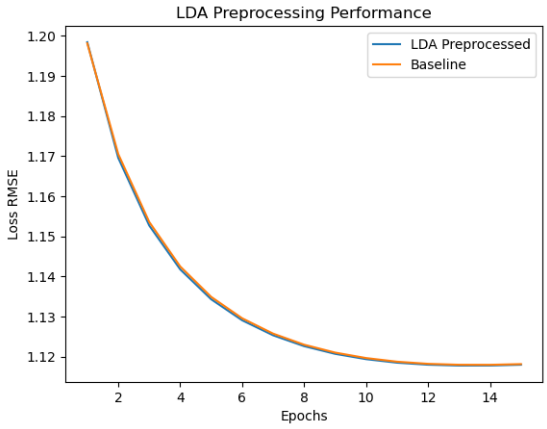

Fig. 10. Weight Initialized vs Baseline

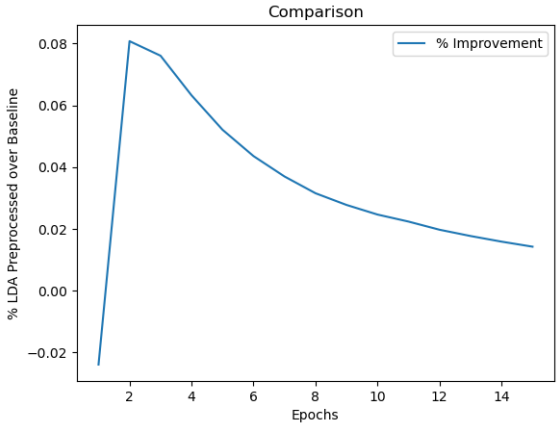

Fig. 11. Weight Initialized vs Baseline Improvement Percentage

As you can see in these topic distributions the LDA analysis unveils a clear theme for around half of the topics generated (5 out of 10). When running the model instead of a smaller number of topics, each topic becomes more coherent and clearer so we decided to go with 5 topics in this experiment rather than 10. As you can see the distinct topics showcase above demonstrate a diverse range of user interests and engagement patterns on the Amazon platform, spanning across gaming, general products, toys, and health-related categories.

As you can see in the figure.10 above the performance evaluation, the comparison graph between the LDA preprocessed model and the baseline model serves as an illuminating visual. The LAD preprocessed model exhibits a slight increase in performance in just a few epochs of training, the magnitude of improvement appears modest, warranting a closer examination. The initialized weights in this comparison show an initial higher loss value compared to the baseline, this unexpected divergence from the expected trend. This discrepancy prompts an exploration into the nuanced dynamics of weight initialization.

This graph in figure.11 demonstrates the performance increase of the LDA preprocessed model with weights initialized

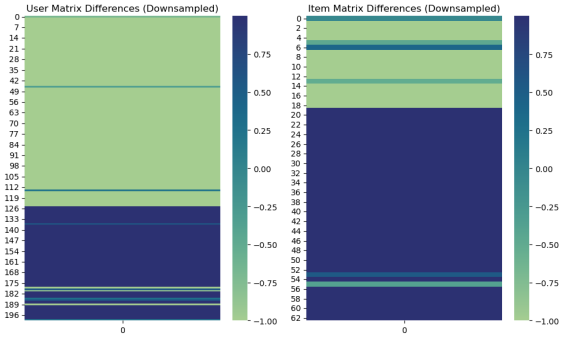

Fig. 12. Weight Initialized vs Baseline Percentage Improvement

to the baseline. As you can see the LDA preprocessed model showcases an incremental improvement, the modest percentage hike compared to the baseline indicates a nuanced impact.

Finally, the heatmap in figure.12 is revealing the embeddings' transition from initial to final weights encapsulates a visual narrative of the model's evolution. As you can see in the figure above, many of the final embeddings do not match the initial embeddings set by the LDA. In the figure above if the embeddings weight didn't change then that value would be 0.00, which you can see only a few embeddings in this downsample image are present. While if the value is -1 or +1 meaning that there was a significant change from the initial weights to the final weights.

C. Topic Mining Exploration

As explained previously, one of the issues with topic generation and building a vocabulary is keeping named terminologies within the model while excluding misspellings. One way this was combated in our analysis of the model was by removing infrequent and frequent terms from the categories. Without this, categories looked like this:

- 'reviewloading', 'replayableness', 'reviewfinally', 'owndriving', 'hotrods'
- 'the', 'and', 'to', 'it', 'you'

Despite this, however, when we do end up with some strange categories when the model is fed with just games:

- 'standby', 'consoles', '8kb', '500gbs', 'powerbrick', '450gb', 'uninterruptable'
- 'immer', 'dieses', 'ich', 'nicht', 'einen', 'ssig', 'szlig', 'keine', 'wie', 'sein', 'wertlos', 'dargestellt', 'lange'
- 'game', 'like', 'games', 'just', 'play', 'fun', 'good', 'great', 'really', 'time', 'graphics', 'story'
- 'zoids', 'actionplayers', '2004publisher', 'moonbay', 'hiltz', 'atariplayers'
- 'halo1', 'sepiroth', 'draugr', 'zigzagoon', 'kijuju', 'swaine', 'dima', 'turians', 'merril'

Despite removing common words, we still end up with a category of very broad terms. Strangely, when examining the distribution of topics for these words, they tend to have the strongest correlation with their category; that is, words such as 'game' and 'like' tended to have the highest normalized distribution in the 3rd category, while words in the other 4 categories tended to have smaller normalized distributions. This remains the case no matter how many times the model is run, although the other categories tend to change a bit. Looking at this, it appears the topics mimic the semantic usage of the word; the first category is related to hardware aspects, the second relates to non-english words, the third are general, non-descript terms, the fourth is older games and the fifth represent modern titles.

When performing LDA on multiple categories, i.e. Toys and Games, Android Applications, and Health, we get the following categories:

- 'protein', 'vitamins', 'vitamin', 'supplements', 'capsules', 'magnesium', 'dosage', 'pills', 'whey', 'garcinia', 'digestive'
- 'game', 'app', 'games', 'play', 'fun', 'kindle', 'graphics', 'playing', 'played', 'quot', 'puzzles', 'screen', 'batteries'
- 'candistry', 'wgu', 'evi', 'poptropica', 'ginkgold', 'mpr', 'prosoft', 'royall', 'smugmug', 'unfollowed', 'unfollows', 'bbn'
- '3100mah', 'vaping', 'rcr123a', 'eagletac', 'intellicharger', '800mah', 'ld715', '880mah', 'cr123', 'evic', '8206', 'r2g', 'i4'
- 'sonicare', 'lancets', 'lancet', 'dentures', 'flossers', 'flosser', 'polident', 'lancing', 'flosses', 'flexcare', 'plackers', 'fixodent', 'brushpicks', 'airfloss', 'floss', 'threaders', 'delica'

Here, we notice that the categories fall back into the original separations imposed by amazon; categories would either be entirely health, or entirely games. There are some categories that can be described as shared, however - category 4 appears to revolve around battery life, which is related to any electronic device and therefore a common overlap between these categories. Once again, there is a noticeable pattern of having a very nondescript category, i.e. category two.

VII. Conclusion

Overall, it appears that LDA does not have a significant impact on the overall performance of matrix factorization. Our initial hypothesis of combining categories of items leading to cross-category topics is proved true, however, showing that LDA is able to draw new connections between categories that would otherwise not be seen invites the possibility of further exploring the effects of topic mining on broader datasets.

For future studies, better resources should be used in order to find better parameters. Additionally, it would be interesting to explore ways to narrow the vocabulary before applying the LDA; as observed, many words within the categories still contained many of the issues of misspelling and removal of spaces, and may have contributed to there being one specific category of generic words.

VIII. Responsible contents

Responsibilities were discussed during regular meetings, and work was divided evenly between the two components. John worked on exploring topic mining with LDA before we

implemented it into the model, as well as the model initializing weights based off of LDA inverse document frequencies, while Nicholas worked on matrix factorization and parameter tuning as well as implementing the model with LDA as a regularizer. Each partner was responsible for analyzing and generating figures for their own representation of the model.

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.