**RoSE: A Broad Approach to Recommending Systems**

By Shawn Oppermann, Hannah Laper, and Sanaz Fecharaki

*Introduction*

The average person encounters at least one recommendation system everyday. Algorithms recommend new music to listen to, new shows or movies to watch, new content to scroll, and products to buy. In this paper, we explore how content data and user data are managed to create recommendation systems that enhance the user experience. With streaming services becoming the main source of viewer entertainment, recommendation systems are becoming easier to integrate, affecting what a user sees first on their page, the order of sub contents, and the order of their titles. With access to user interaction data and a large sample space of movies, recommendation systems are becoming more vital and more complex. For the streaming service Netflix, “80% of stream time is achieved through Netflix’s recommendation system” (Chong, 2021). However, evaluating the effectiveness of a recommendation system before it is deployed is incredibly difficult, as its effectiveness can really only be evaluated by user satisfaction or general intuition. This paper introduces RoSE (Recommendation System Exploration) as a way to quickly search through types of models and review example recommendations provided.

*Objective*

This paper will explore the theory and implementation of both content and user based recommendation systems. Different content-based approaches will be implemented and compared, and a user based recommendation system will also be implemented. Models will be presented though a user friendly dashboard, that allows users to compare and combine different recommendation approaches.

*Dataset*

Two datasets will be implemented for analysis. From Kaggle, we used movies\_metadata.csv. This is a dataset extracted from IMDB.com, a movie database website. The dataset consists of 24 columns and 45,466 rows; the columns used for analysis are as follows:

* Id: id assigned to movie
* Genres: listed genres of movies
* Imdb\_id: id given to movie by IMDB
* Original\_title: original title of movie
* Title: title of movie
* Production\_companies: production companies of the movie
* Production\_countries: countries where the movie was produced
* Spoken\_languages: different languages the movie is available in
* Tagline: A short string that is that explains the movie; likely to be seen on the movie poster
* Overview: Summary of the movie

This dataset will be used for content-based recommendation systems.

We also used data from MovieLens, specifically the MovieLens 25M dataset. This dataset consists of six csv files each containing different data. The csv files used in this project are as follows:

* Movies.csv: This dataset contains a movieId, titles, and genres
* Tags.csv: This dataset contains userId, movieId, tag, and timestamp. This dataset records string type tags that users imputed for different movies
* Links.csv: This dataset contains movieId, imdb\_id, and tmdbId. This dataset is useful for merging with the movies\_metadata.csv dataset using the imdb\_id column.
* Ratings.csv: This dataset contains userId, movie\_id, ratings (from 1-5). This dataset will be used for the user-based recommendation system.

To assist with the content based recommendation system a new dataset was created from merging the movies\_metadata dataset with the MovieLens dataset to create a dataframe called content\_df. This dataset consists of 62, 423 unique rows of movies and 10 columns. The columns are as follows:

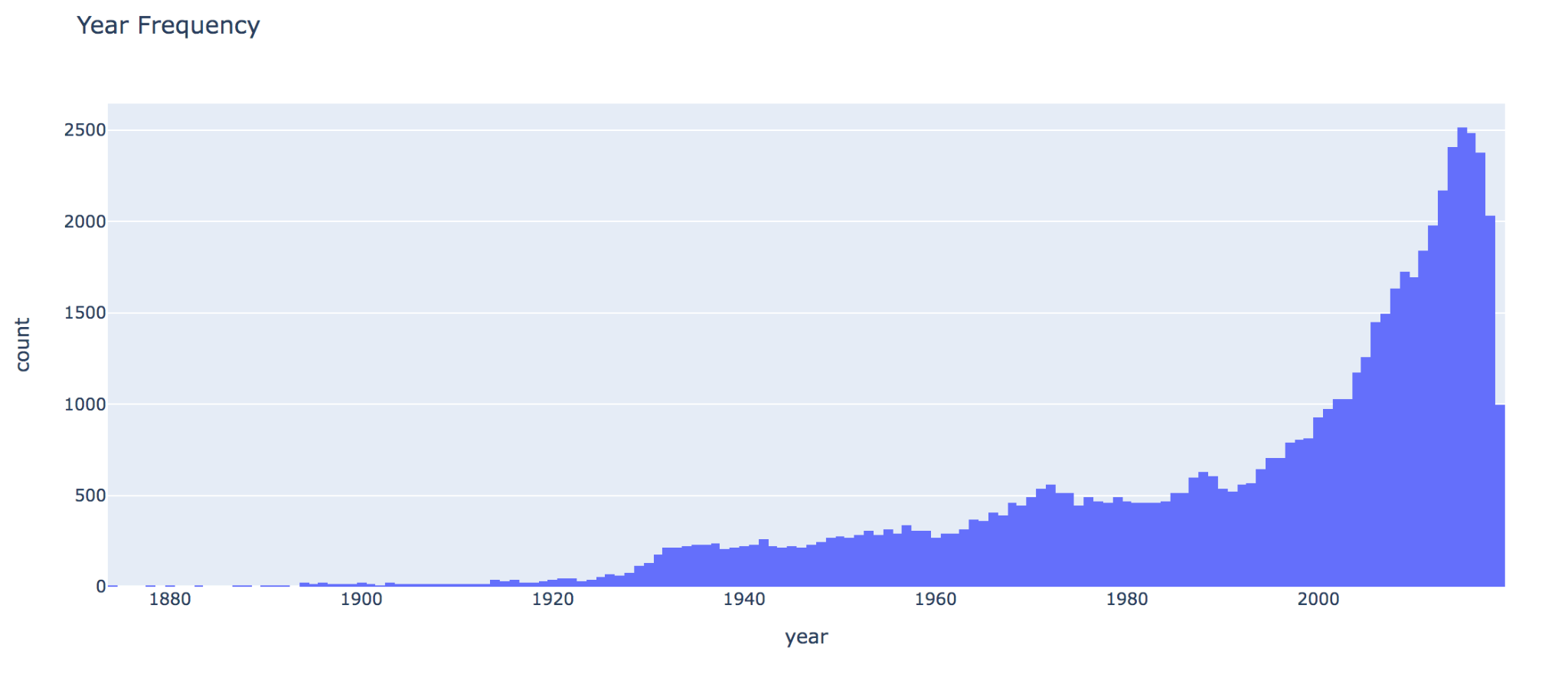
* movieId: id value given by MovieLens
* Imdb\_id: id value given by IMDB
* tmdbId: id value given by another database (not used in this analysis)
* Overview: summary of the movie
* Production companies:
* Tagline: A short string that is that explains the movie; likely to be seen on the movie poster
* Title: title of the movie
* Genres: genres of the movie
* Year: year the movie was released
* User\_tag\_list: short strings submitted by different viewers of the movie

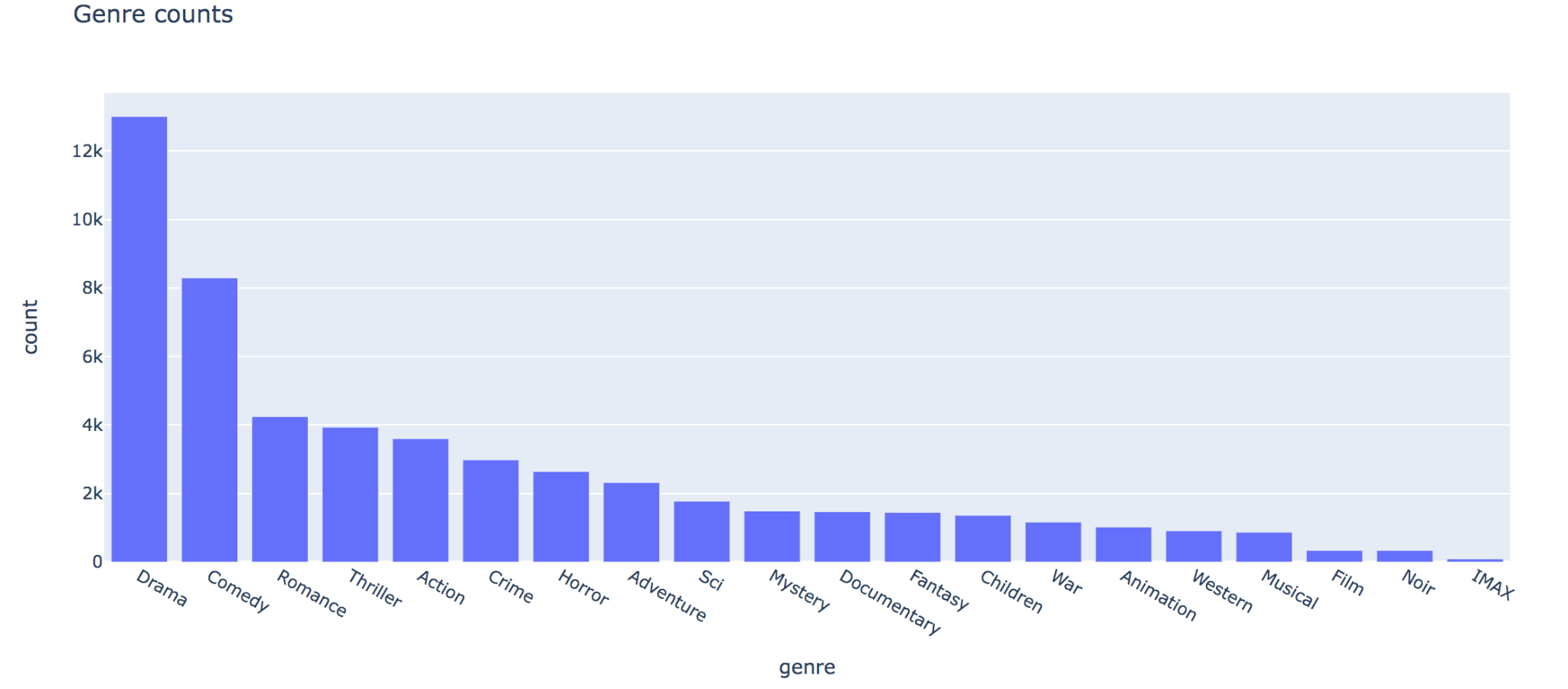
*Preprocessing*

To create a corpus, the columns genres, user\_tag\_list, overview, and tagline were used. For each column, all strings were converted to lowercase letters. The overview column is the largest part of the corpus, so rows without an overview were dropped from the dataset. For user\_tag\_list, a function was implemented to lemmatize each user tag, remove stopwords, and lemmatize. Then duplicate user tags were removed leaving a list of unique user tags. Functions were created to remove special characters, remove stopwords, and lemmatize. These functions were applied to the overview and tagline columns. Finally, the columns were combined to create a corpus for each movie which is stored in the corpus column of the content\_df.

*Data Exploration*

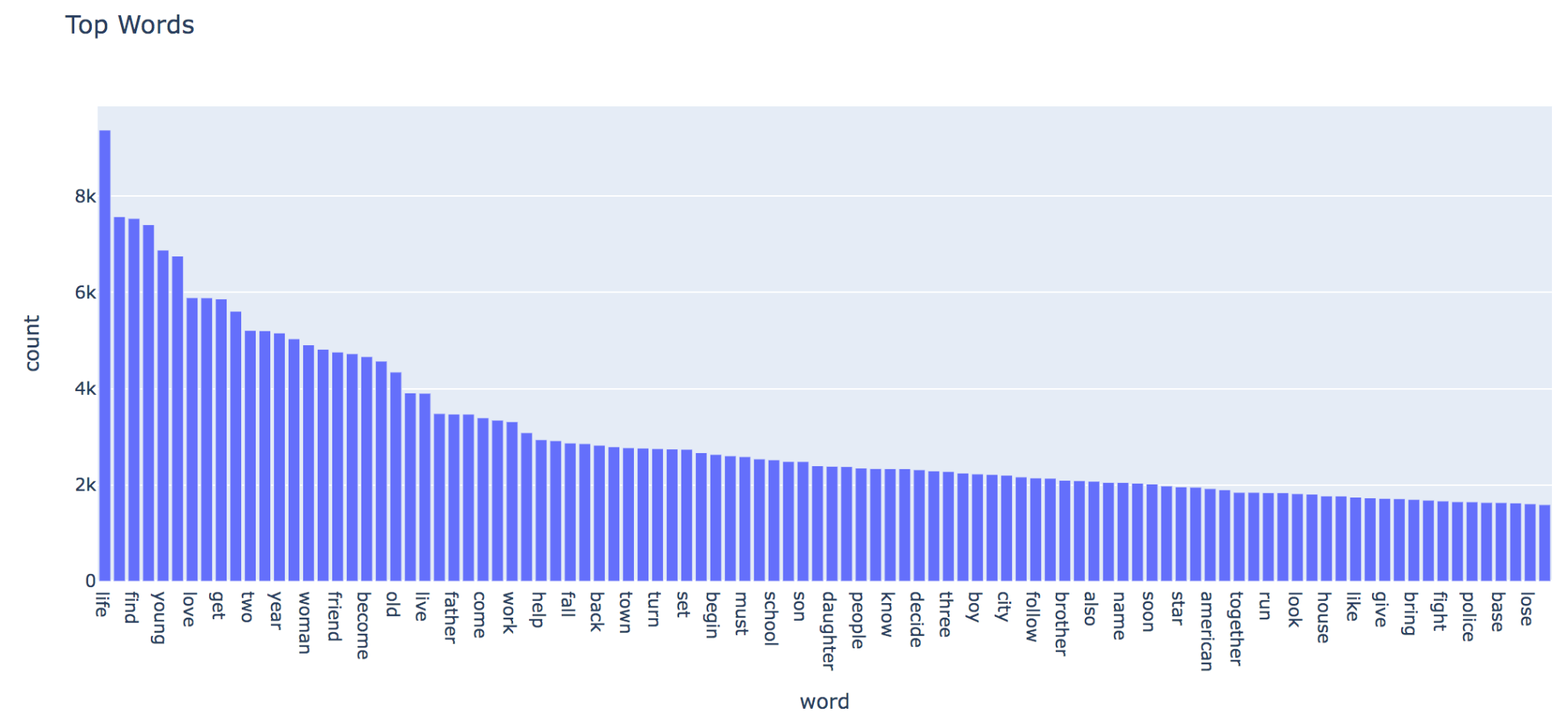
To get a general understanding of our data, we started with some basic data exploration. The movies in our dataset range from 1874 to 2019 with 2015 being the most common year as depicted in the image below.

There are 20 different genres used to classify the movies. For each movie, there can be no genre listed, one genre listed or multiple genres listed. The following image shows the count of each genre.



The most common genres are drama and comedy, while the genre IMAX is quite small and possibly not worth considering. This is important to note when clustering because there may be clusters that are subsets of drama while only one musical cluster.

To understand the overview column, we looked at the distribution of words in the preprocessed overview column. Below are the results



The most common words are life, find, young, and love. One can assume that most dramas usually are about life and finding something whether it be a person, thing, or meaning, and the words young and love typically go with romance. Since drama and romance are among the top genres it would make sense that words usually associated with those genres are the most occurring.

*Related Research*

Recommendation systems can be content-based, user-based, or a mixture of both. With a content-based system, recommendations are based on the product. With a user-based system, recommendations are based on user-item interaction, defining items by the users that like them. Content-based recommendations allow for cold starts: new users without any data. User-based are more fluent and will change the more users interact with the interface (Chong, 2021). For this analysis, content-based recommendation systems are the main focus.

Content-based recommendation systems use unsupervised learning because the movies are not labeled by topic. Specifically, content-based recommendations need a clustering and/or topic modeling approach. The most common topic modeling techniques are Latent Dirichlet Allocation (LDA) and Non-Negative Matrix Factorization (NMF) (George, 2021). LDA is a statistical model that derives latent or unobserved topics in a corpus (Gosh, 2019). NMF is a linear algebraic clustering model. Both LDA and NMF take a preprocessed corpus and transform it into a TF-IDF matrix as an input to determine the clusters. Because both these models are unsupervised it is hard to evaluate their accuracy. Execution time and mean coherence scores can be calculated to evaluate LDA and NMF models (George, 2021). Shini George and V. Srividhya compared LDA and NMF models of restaurant reviews and found that both the execution time and mean coherence values for the NMF model exceeded the LDA model (George, 2021).

The Doc2Vec method is a concept that was presented as another way to vectorize a corpus (Le & Mikolov 2014). Doc2Vec utilizes Word2Vec to build a hyperdimensional space which maps each text to a vector. Using a neural network approach, Doc2Vec provides vector representations of variable-length pieces of text, such as paragraphs, or documents. The vector represents the semantics of the input text. Therefore, the texts with similar meaning or context will be closer to each other in vector space than unrelated or less-related text. (Sharaki, 2020)

Evaluating content-based models is vague because you are working with unlabeled data. One way of evaluating content-based models is by inspecting how well a model predicts what the user is going to watch (Shani, 2011). However, the purpose of using recommendation systems spans further than simply being able to see what a user will watch. Companies want users to be recommended new materials and expand the diversity of what they currently use/watch so that they will keep using the service. Therefore, to evaluate a content-based recommendation system instead of wasting computational resources to see what a user will predict, one should evaluate user satisfaction with the recommendation device (Shani, 2011).

Evaluating recommendation systems requires having data on users’ reactions to the recommendation provided. This is not very realistic in this case, since we do not have a user base ready for testing. In this case, the best approach is to try various models and gauge the sensibility of the results personally. With RoSE, it is possible to quickly compare the recommendations of different models, and decide which model works best in the current context.

*Analysis Model*

Based on the research above, we wanted to explore different ways to perform grouping, vectorization, dimensionality reduction, similarity, and scoring of selections from the MovieLens dataset. Recommendations are based on genre, plot, and user ratings. Similarly to how the optimization algorithm, grid search, allows you to find the best parameters for a problem; our pipeline would allow you to compare various recommendations across different pipelines. The first algorithm of the pipeline is the grouping of the data.

Grouping can either be done by genre, movie clusters created through gensim’s KMeans clustering on the corpus, or no grouping at all. Once the grouping is decided, a separate model is created for each group, i.e. a model for only action movies,a model for only drama movies, etc.

We perform topic modelling, or essentially dimensionality reduction, through NMF or LDA. For similarity scores we select manhattan distance, euclidean distance, or cosine similarity. In most cases, we choose cosine similarity. Lastly, if the initial selection includes multiple movies, the user can combine the similarity score across multiple movies using the magnitude squared, sum, or max or the movie scores. The vectorization, dimensionality reduction, similarity, and scoring all use modules from the gensim library.

For user-based recommendation systems, we use the ratings.csv file to create a user-item interaction matrix. Then we will use cosine similarity to find similar users, and optionally, NMF to reduce the number of dimensions.

*Outcomes*

In initial testing, we compared an LDA model with TF-IDF vectors, an NMF model with TF-IDF vectors, and a Doc2Vec model. Doc2Vec is technically a vectorizer, but further topic modelling is not necessary. There is no grouping involved for this test. We ran all three models for the movie *Toy Story,* an animated children’s film*,* and found the following results:

| LDA Recommender | NMF Recommender | Doc2Vec Recommender |
| --- | --- | --- |
| Winter Light Nattvardsgästerna | Madagascar Escape 2 Africa | Flintstones |
| Craft | Sinbad Legend of the Seven Seas | Bolt |
| Finder’s Fee | Alice in Wonderland | Flintstones in Viva Rock Vegas |
| K PAX | Mulan | Toy Story 3 |
| Finding Nemo | Bug’s Life | Geri’s Game |
| Naked Lunch | Flushed Away | Stuart Little |

NMF and Doc2Vec perform very well, recommending other animated children’s movies. Therefore recommending within the same demographic and area of movies. LDA struggles to find the right audience, as the movies contain overviews with themes such as ‘meaningless existence’, ‘patients in mental hospitals’, and ‘addictions to substances.’ These types of movies most likely will not be wanted by the same demographic that watched a G-rated children’s movie.

Next, we try the three same models with the movie, *The Matrix,* a science fiction, action packed movie for older audiences. The results are as follows:

| LDA Recommender | NMF Recommender | Doc2Vec Recommender |
| --- | --- | --- |
| Short Cuts | Star Wars Episode II Attack of the Clones | Avengers |
| ​​Equilibrium | Children of Men | Watchmen |
| Johnny Mnemonic | Dracula Untold | The Matrix: Revolutions |
| Lord of the Rings The Return of the King | Star Wars Episode III Revenge of the Sith | Animatrix |
| The Blade Master | Iron Man | Blade Runner |
| Tango Lesson | Avatar | Batman Begins |

With this movie the LDA improves, recommending some science fiction and fantasy movies, however NMF and Doc2Vec still perform better. NMF is able to recommend science fiction, action, and fantasy movies. Particularly impressive is that Doc2Vec recommends *Animatrix* and *The Matrix: Revolutions* movies directly based off of *The Matrix*, as well as other science fiction and action movies.

Further testing involves comparing three NMF models grouped by nothing, genre, and K-means clustering, and comparing using a user-item matrix with or without NMF. The outcomes of which can be found in the jupyter notebooks included with the paper.

*Conclusion*

RoSE allows us to test both content-based and collaborative approaches to provide movie recommendations, quickly testing different options for our models. For vectorization, we can choose from: BOW, TF-IDF, Doc2Vec. We also test grouping our data by genres (can have multiple genres per movie) or K-Means clusters (only one cluster per movie). For dimensionality reduction we use both (Non-negative Matrix Factorization (NMF) Topic Modelling and Latent Dirichlet Allocation (LDA) Topic Modelling methods. The distance metric can be cosine similarity, manhattan distance, or euclidean distance, though cosine is used as the main measure of similarity between movie plots. When creating a recommendation based on multiple movies, we take the sum of squares (square magnitude) of the score against each movie.

Through this approach, we have generally found NMF to be an effective topic modeller, and Doc2Vec to be an effective vectorizer for the MovieLens dataset. The performance of each approach can vary widely across different contexts, so it is encouraged to use an approach like RoSE to quickly determine which model best suits your data.

*References*

Chong, D. (2021, September 24). *Deep dive into Netflix's Recommender System*. Medium. Retrieved December 1, 2021, from https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48#:~:text=80%25%20of%20stream%20time%20is,is%20a%20highly%20impressive%20number.

Dilmegani, C. (Updated 2021, November 17)‌. *Recommendation Systems: Applications, Examples & Benefits.* Retrieved December 3, 2021, from https://research.aimultiple.com/recommendation-system/

George, Shini & Vasudevan, Srividhya. (2021). Comparison of LDA and NMF Topic Modeling Techniques for Restaurant Reviews.

Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. In *Recommender systems handbook* (pp. 257-297). Springer, Boston, MA.

Sohom Ghosh, & Gunning, D. (2019). *Natural language processing fundamentals : build intelligent applications that can interpret the human language to deliver impactful results*. Packt Publishing.

Dipanjan Sarkar. (2019). *Text analytics with Python : a practitioner’s guide to natural language processing*. Springer Naure.

Le, Q., & Mikolov, T. (2014, June). *Distributed representations of sentences and documents*. International conference on machine learning (pp. 1188-1196). PMLR.

Sharaki, O. (2020, July 10) *Detecting Document Similarity With Doc2vec, A step-by-step, hands-on introduction in Python.* Retrieved December 3, 2021, from https://towardsdatascience.com/detecting-document-similarity-with-doc2vec-f8289a9a7db7