Recommendation System for Jokes

Efthymios Prapas, Team C

January 28, 2024

Abstract

In this short paper, we explore the topic of recommender systems and we complete a short review of the available methods. We also assign review some of the advantages and disadvantages of this paradigm, hoping to position it among other machine learning techniques. Consequently, we implement several of the described methods in order to compare their performance and reach conclusions regarding their effectiveness when used on a dataset of jokes.

1 Introduction

Modern applications provide access to a stream of information that is difficult to handle by a single person. The most characteristic example is the Internet itself; from its creation, there has been an effort to categorize and rank web-pages according to their perceived relevance to the user among other criteria. Most search engines in order to restrict the surveyed webpages had an explicit query. The most important advance in the field of search engines is arguably the PageRank algorithm, which provides efficiency, consideration of all possible relevant webpages and query-specific results.

Recommender systems ”are efficient tools for filtering online information” (Roy, Dutta, 2022). Their usual use cases apply when information is needed for a set of items which are available to a set of users. According to another, more concrete definition, a recommender system is ”any algorithm that uses large datasets to identify similarities and recommend decisions to users” (Goncalves-Sa, Pinheiro, 2023). Specifically, when there is the possibility of explicit feedback from the users, this feedback can be used to recommend items to other users (Murphy, 2022). In order for this to happen, the algorithm of the recommender systems may segment users according to their tastes, items according to their similarity or both (Goncalves-sa, Pinheiro, 2023). This functionality leads to many applications in the fields of movies, books, tourism (Roy, Dutta, 2022). Their usage is not restricted to consumption goods though; they can also be used in social media platforms (Youtube, Facebook suggested content) or even scientific papers (Beel et al., 2016).

The question then arises, why would we need recommender systems; would it not be easier to just adapt search engines over narrower fields of the Internet? The answer to this consists of many parts and we shall touch upon some of them. First of all, recommendation systems can suggest items even without user input. This circumvents the need for an explicit query by the user. It thus seems natural to consider recommender systems as a form of ”soft-query”. The user might not make an explicit query but the recommendation system might still make a useful recommnendation upon which the user will act. Additionally, the user does not need to specify what he wants. This gives flexibility to the system to suggest similar items; compare this to a common search query which would yield items restricted by the specifics of a query, while a desirable item is knocked down in priority because the user is using a synonym (extreme example). Finally, recommendation systems can personalize suggestions. While a personalized version of the PageRank algorithm would be inefficient, recommendation systems circumvent that difficulty and offer suggestions tailored to each user.

We introduced the topic of recommender systems with a comparison to the perhaps more familiar topic of search machines. This should not deceive us with regard to their age, since the first application of these systems was in 1998 by Giles et. al, for indexing of citations. There were also patents related to collaborative filtering, an algorithm for recommendations, in 1989 (Goldberg et. al, 2001). We will now proceed to adapt the framework of recommender systems to a dataset for jokes, part of Jester. Jester is a joke recommendation implementation that uses the Eigentaste algorithm (Goldberg et al, 2001). In order to circumvent the cold start problem (more on this later), the researchers used a dataset of 100 initial jokes and collected ratings of them over a period of 4 years (1999-2003). We will use the same dataset to implement some of our less exotic methods. It should be noted, that the dataset is relatively dense, since we are using the dataset of 48483 each of whom has rated at least 36 jokes.

2 Recommendation Systems Review

2.1 Roadmap

First, we will analyze the different approaches that one can take to recommendation systems; this includes both a classification and a discussion of the main principles for each category. Then, we focus on the advantages and disadvantages of those systems, as well as their evaluation methods. Capping off our theoretical part are other approaches that can be used to generate recommendations, such as nearest neighbour algorithms as well as a short review of using collaborative filtering for jokes.

2.2 Categorization of Recommender Systems

There are several ways to distinguish between recommendation systems. A major one is based on the type of information that they are given as input. Recommender systems can take in information about the features of the items and make suggestions based this information. This is termed content-based filtering. In a 2016 literature survey (Beel et al.), it proved to be the most popular with the research community. Briefly, item data and characteristics are used to aggregate items into profiles (Roy, Dutta, 2022) and then the interactions of users with these items are used to build user models (Beel et al., 2016). The features that can be used are countless; some simple examples in the model case of movies are the text titles, text description and with further processing, movie posters (Murphy, 2022). In the field of scientific paper recommendations, features such as the abstract, foreword, author and citations can be used to provide a categorization (Beel et al. 2016). In most of these cases, the algorithm to estimate similarity between texts was TF-IDF. TF-IDF is an algorithm that is accepted as a measure of how important and specific to a text a word is to a document (Murphy, 2022). There can be other methods of semantic analysis too (Lops et al., 2011) but TF-IDF embodies several appropriate assumptions when building a keyword vector space (Lops et al., 2011) Finally, content-based filtering is used more in news and publication recommender systems (Roy, Dutta, 2022). This is probably because the quality and standards of the content are more important for the final user experience than in movies, songs etc.

Another approach that is used abundantly in recommender systems is collaborative filtering. In certain ways, this is an antithetical paradigm in relation to content-based filtering. The core insight behind is that users will want to interact with what "similar" users have already interacted with in a positive way. This is used to aggregate user profiles and generate recommendations based on history of past users. For this category, it is critical to evaluate similarity of users (Roy, Dutta, 2022). While in the simple setting, this "clustering" of users would be done based on their likes and dislikes (implicit or explicit ratings), we can add contextual information about the users in order to avoid the problem of lack of information at the start of the implementation (Murphy, 2022). In collaborative filtering, there are 2 paradigms. The first one is that of neighborhoods, where we focus on "user-user relationships" (Koren, Bell, 2011). The other approach is that of latent factor models, where both the users and the items are embedded into the same item space (Koren, Bell, 2011).

Finally, there is a 3rd major category, that of hybrid systems. Hybrid systems incorporate recommendations from various other systems in order to suggest items.Often, hybridization consists of using a content-based technique to improve results of collaborative filtering (Roy, Dutta, 2022). This is also the case for scientific papers recommender systems, as exemplified by the TechLens proprietary algorithms (Beel et al., 2016).Some of the hybridization approaches are meta-level, feature-augmentation, feature-combination, mixed hybridization, cascade hybridization, switching hybridization and weighted hybridization (Roy, Dutta, 2022). We explain some of these approaches in our own words. Meta-level hybridization takes place when we use a cascade of differing types of recommendation systems, where the input of one is the output of the other. The most simple cases are weighted hybridization, where the different predictions are assigned weights and then combined according to their weights, as well as switching hybridization, where depending on the context (for example absence of features, absence of feedback), we use either a collaborative filtering system or a content-based system (Roy, Dutta, 2022). These techniques exemplify what Burke and Ramazani (2011) said about the creation of hybrid systems: "...adapting an algorithm for one recommendation type to accept a knowledge source more typically associated with another type.". That is also the case here.

There are also some other categories, that are either less popular or credibly can be dealt with as subcategories of the aforementioned categories. These include stereotyping (hard grouping of users based on few characteristics), co-occurence of items as well as graph based methods (Beel et al., 2016). We do not delve more into these categorizations, as their mathematical formulations would make them fall into the framework of one of the previous categories.

We also have other categorizations of recommender systems, based on the knowledge that is needed to make a recommendation. This is an interesting approach, since it abstracts away from the details of each algorithm. Broadly, these recommender types still consist of content-based and collaborative recommendations (Burke, Ramazani, 2011), which have a differing focus. Content-based filtering is based on the individual user, while collaborative filtering matches "an individual knowledge source with a social knowledge source of the same type" (Burke, Ramazani, 2011). There is also the category of knowledge recommendations, that are based on any field of knowledge more concrete than the field of the items themselves (Burke, Ramazani, 2011). At this point, it is also appropriate to notice the 2 different types of feedback that can be given to a system. The best case is explicit feedback. A user tries an item and returns an explicit rating. This yields a lot of information and is the ideal case scenario. Nonetheless, not all users are invested in this (Roy, Dutta, 2022) and this can lead to sparsity of the end results (Appaji et al. 2023). This can happen in some contexts, especially when the users are few and the items are many, such as in academic papers recommendations (Beel et al., 2016). The other approach is that of incorporating implicit feedback. The main idea is that the values that are not filled in are not "missing at random" but rather represent a conscious choice of a user not to rate this item (Murphy, 2022). This yields more data that can be used to estimate preferences. However, this approach can limit new suggestions in collaborative filtering.

2.2.1 Architecture of Recommendation systems

There are several steps involved in recommendation systems. The most simple, non-peer-reviewed viewpoint that we found is by Panarin (2023), who claimed that there are 2 steps to recommendations: Candidate generation and Scoring. Candidate generation involves creating possible suggestions, while scoring involves pruning the candidates and sorting them. In the Google course suggested in our course, there is an additional step, called re-ranking. Essentially, this step is to apply some constraints that might appear in the query or to emphasize newer products. For example, some items might be explicitly banned by user (parent-control possibly) and some probabilistic mechanism for new relevant items appearing would also take place here. For content-based filtering, there are 3 essential components to the process. First is the content analyzer, who basically extracts features from the description of items (Lops et al., 2011). Second is the Profile Learner, which is essentially the part that corresponds to candidate generation and finally there is also the Filtering Component which is not dissimilar to ranking as described above. Of course, there are pipelines for the user to provide feedback (Lops et al., 2011).

2.3 Advantages and Disadvantages

Each of the above described techniques has its own specific strong points but also drawbacks and we will describe the bulk of them. First of all, some problems that plague all recommender systems.

When we have items that are practically the same but have a similar name, there can be issues of synonymy. Namely, the recommender cannot take into account that the items are similar in collaborative filtering. In content-based filtering, having the same item under two names can also cause issues. This means that the recommender becomes less accurate for these items (Roy, Dutta, 2022). Another problem that appears often is that of data sparsity (Roy, Dutta, 2022). There are several ways to deal with this, one of them is using Singular Value decomposition in model-approaches (Roy, Dutta, 2022).

The cold-start problem is important for many recommender systems (Roy, Dutta, 2022); there needs to be some "social" information at the start in order to start generating recommendations, especially for collaborative filtering (Burke, Ramazani, 2022). Content-based and collaborative filtering techniques are impacted differently though. For example, when we have new users, we cannot implement collaborative filtering due to the fact that the user can not be assigned similar users without making choices. For content-based filtering, a new item can be assigned based on its features to a group of items and then we can recommend it but without feedback the items in the group become less reliable. Nonetheless, there are ways to deal with it such as implicit ratings depending on the features of interactions (time spent etc.). A more important appearance of cold-start problem in content-based filtering occurs when we have a new user and we have nothing to recommend to him because we cannot build yet a user-profile (Appagi, 2023).

Another interesting dilemma that affects all recommendation systems is raised by Murphy (2022) and is called the exploration-exploitation tradeoff. Intuitively, at each point the recommendation systems has 2 somewhat conflicting goals. In the short-term, it wants to provide the best recommendation possible. In the long-term however, it would like to get more explicit feedback from the users in order to generate better recommendations in the future. This is especially so when the system is "uncertain" about its recommendation (Murphy, 2022), so it might make sense to suggest a more exploratory item. All recommender systems need to navigate this dilemma which is attributable to the fact that the data they train on are based on their output (Murphy, 2022). The point is that a recommender that is too focused on suggesting the best match might fail to uncover items of interest for the user in the long-term.

Regarding specific types of filtering, content-based filtering has some advantages. First of all, filtering focuses on individualized suggestions. Hence, we can build a profile only dealing with the ratings of one user. This individualized approach is core to pure content-based filtering and is especially helpful in alleviating sparsity problems (Appaji, 2023). Collecting multiple side features can also help classification of items (Murphy, 2022). Additionally, this individualized approach helps suggest specific items to users which would have been drowned out in the case of collaborative filtering. This yields many advantages which are manifest as drawbacks of collaborative filtering.

We now focus on some of the specific drawbacks of content-based filtering. First of all, there needs to be a certain knowledge of which features are important for item classification (Beel et al., 2016). Also, the algorithm is by nature relatively conservative and it suggest already popular items among groups of interests of the user, leading to few "lucky" finds (Beel et al., 2016). Finally, we have that unique items could provide difficulties in generating recommendations.

Collaborative filtering provides many advantages. We mention 3 as mentioned in (Beel et al. 2016). First of all, pure collaborative filtering is content-independent, which means that there is no need for processing items. Second, since ratings are done by humans, there is a quality assurance to a certain degree which is not the case with content-based material (lumped together in groups). Finally, recommendations can be more varied since they are also based on user similarity; we can discover niche topics that interest similar users.

Nonetheless, there are several drawbacks to this approach. Due to the fact that recommendations are for groups, the scalability and processing times for collaborative filtering are worse (Beel et al., 2016). The latency problem is when new items fail to get recommended to the users (Roy, Dutta, 2022). This is of course because they have few reviews and this does not occur in content-based filtering. There is also the issue of "grey sheep" (Murphy, 2022), namely users that do not fit conveniently into any category. Recommendations for these users can suffer from inaccuracies.

Finally, there are some more topical issues that are of societal relevance too. Collaborative filtering overindexes on already popular items (Panarin, 2023). This development has led to critiques that such systems essentially manipulate opinion (Goncalves-sa, Pinheiro, 2023) and homogenize it (Beel et al., 2023) according to other users. This can lead to political polarization (more discrete political groups) among others (Goncalves-sa, Pinheiro, 2023). Additionally, there are also privacy concerns due to the efficiency of these algorithms. This is especially true when they are used by industry giants who have access to a lot of side features (Goncalves-sa, Pinheiro, 2023). A last point of pain is the manipulation of ratings. While content-based filtering might fall pray to manipulation from item-generators, collaborative filtering is much more sensitive to it. The problem is referred to as "shilling attack" (Roy, Dutta, 2022) or "blackguards" (Beel et al., 2016) but the point is the same. Manipulative users might downvote or upvote items dishonestly, leading to a real change of recommendations not only for them but also to other users. It is crucial that fake ratings/users are deleted as soon as possible (Roy, Dutta, 2022).

2.4 Evaluation of recommender systems

There are several measures that can be used to evaluate differing recommender systems. In systems where feedback is binary, the usual applicable metrics of machine learning apply, namely accuracy, precision, recall, F1-score and others (Herlocker et al., 2004). These scores should not be all used in different contexts because in different contexts there is differing emphasis on false positives and false negatives (precision in the negative or positive class namely). In our case, with continuous metrics, we can either decide to discretize our results, encode them categorically or use continuous measures. We mainly adopted the MSE and the RMSE as measures. In the original Jester paper, the authors use Mean absolute error and normalized absolute mean error. We tried implementing it too for reasons of comparison, though with limited success.

2.5 Other Possible avenues (KNN, embeddings)

K-nearest neighbour is another case where we have the ability to implement recommendations. Essentially, this technique showcases instance-based learning and it is a technique to cluster (Vlachavas et al., 2020). The main insight behind this algorithm is the fact that if our data is continuous, we can assign to them a distance based on how much their individual features deviate. This then can be used to produce clusters using either k-means clustering or as in the present case, having the designation of the nearest neighbours. Hence, if there are labelss to be learned, in an k NN algorithm the class mode of the k nearest neighbours would be our prediction. In our case, we can use this to get some neighbours, weigh their features appropriately and suggest a joke. We make the choice to suggest the most popular jokes among the nearest neighbours.

There are 2 crucial implementation issues in kNN. THe first is the number of points. 1 neighbour is rarely enough and in order to be robust to outliers, severals are needed. However, we cannot have the value of k increase as much as we would like and expect good performance. In the case of k larger than the number of datapoints, we are sure that the algorithm will assign to all points the mode of the class labels (Vlachavas et al., 2020). Hence, a compromise is needed for optimal implementation. The other concern is that of distance. Distance would ideally be a metric, there are various other measures such as cosine similarity and norm-based distances (Vlachavas et al., 2020). It is also critical that normalization takes place since dot products can give too much weight to datapoints with large feature values.

(Based on course materials) Finally, we also approach embeddings. Embeddings are essentially a way of mapping individual items into a geometric usually space that is less complex. This allows us to see their relationship better if the space is 2D, or at least reduces complexity. There are several ways to get such embeddings. One of these ways is collaborative filtering through matrix factorization (Murphy, 2022), though there are also others such as neural networks and autoencoders (Murphy, 2022).

Some of their advantages are the fact that they can be used with discrete and continuous data, they reduce dimensionality and depending on the technique they can catch complex interactions. Nevertheless, they too suffer from cold-start problem (no embedding possible) and high computational costs.

2.6 Recommendation systems and Jokes

Here, we go over the seminal paper of Goldberg et al. 2021 introducing the Jester recommendation system. The source of our data is a website set up so that users rate jokes on a horizontal bar, with a continuous scale. The system is rather refined, since it does not use collaborative filtering blindly but it uses dimennsionality reduction and clustering to reduce computational load before using collaborative filtering. So, a user gives his ratings on some representative jokes, then his vector of ratings is projected into the joke space, where we find the corresponding cluster and thus our suggestion. This specific algorithm, called eigentaste, was compared with nearest neighbour algorithms and just suggesting the most popular joke on average. The eigentaste algorithm showed significant computational gains with respect to the nearest neighbours algorithms, where 80 neighbours were needed to provide a good approximation. Taking the best joke without personalization was less reliable, but still very good and comparable to the other methods (better than 1NN).

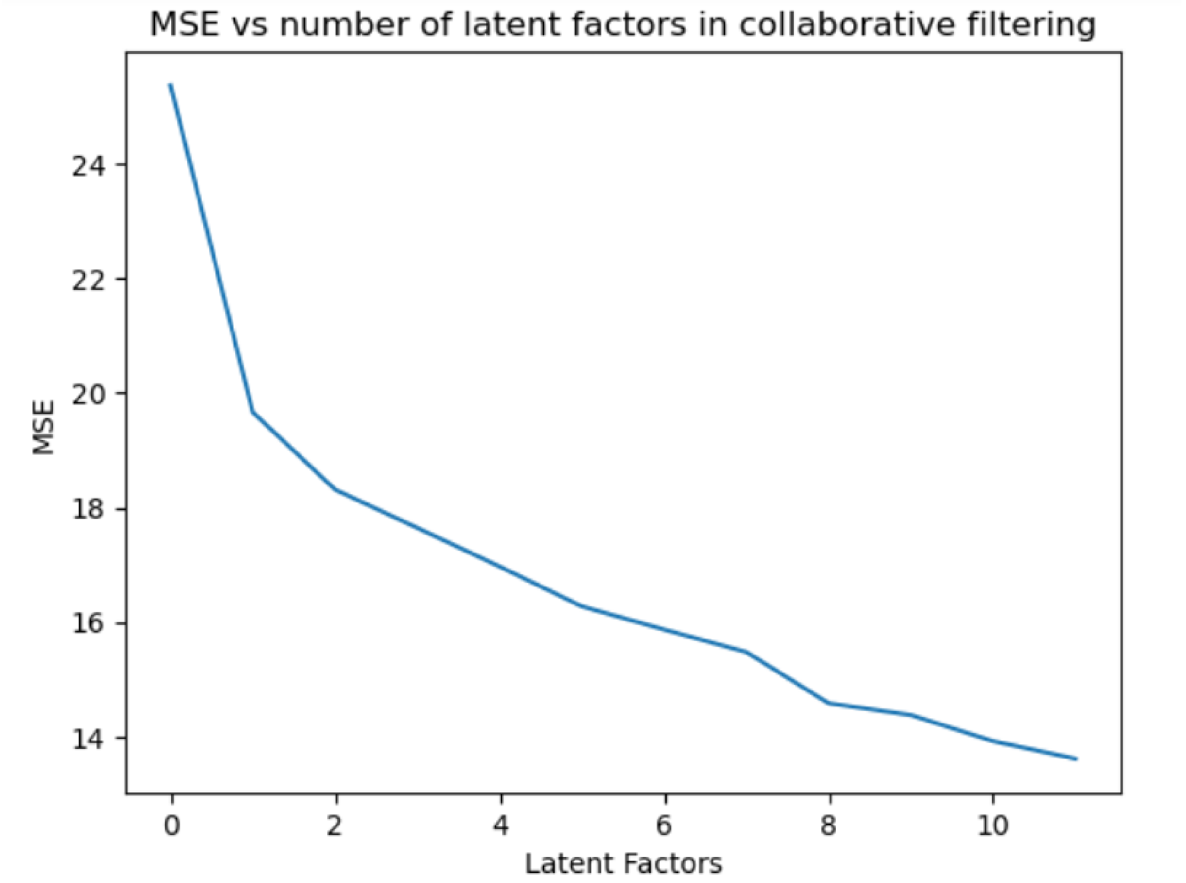

Figure 1: This was the result of adjusting number of latent factors.

2.7 RESULTS

First of all, we try to implement collaborative filtering. The easiest way to do it is as described in the theoretical part. Since our dataset has Nan, we impute with 0 the gaps. We created a function that is called $generate_recommendations$ in order to implement the SVD approximation to our full matrix. Then, we also created a *compare* function which gives us the MSE, given the initial and the number of latent factors. Finally, we also implemented a function to approximate the MSE through repetitive sampling.

The results were clear. The MSE was relatively low and even with 9 latent factors it was below 14. This yields better results that Jester algorithm, though the sample size is a meager 200 users. It should be clear that the larger the matrix, we can expect more latent factors to be needed.

We proceed with the description of our 1 nearest neighbour implementation. The reasons for choosing k=1 will be analyzed in the discussion. The architecture of our program is relatively simple. Given a dataframe with the cartesian product of all the users and the jokes where ratings are assigned to each element, we first create a dictionary that points each user towards his closest neighbour. This is accomplished with the function $nearest_neighbour$. Then, we use this function within our function $recommend_suggestions$ which yields a single recommendation, the best perceived joke of the neighbour that has not already been seen by the user. Here, the results are difficult to evaluate and no methodology has been used towards this goal; similar to nearest neighbour tesselations, without an appropriate test set it is difficult to extrapolate (more on this in the discussion).

Finally, we implemented a neural network approach in Keras. While the code was ready for a large part of the assignment, the necessary data transformation was not trivial and involved reshaping the matrix so that each rating instance is its own row. We trained the model through 10 epochs, wherein the Mean Squared Error dropped, approaching 15 at 30 epochs. The results for are still not ideal, though competitive with the Eigenstate algorithm from Goldberg et al.(2021). Ideally, we would test

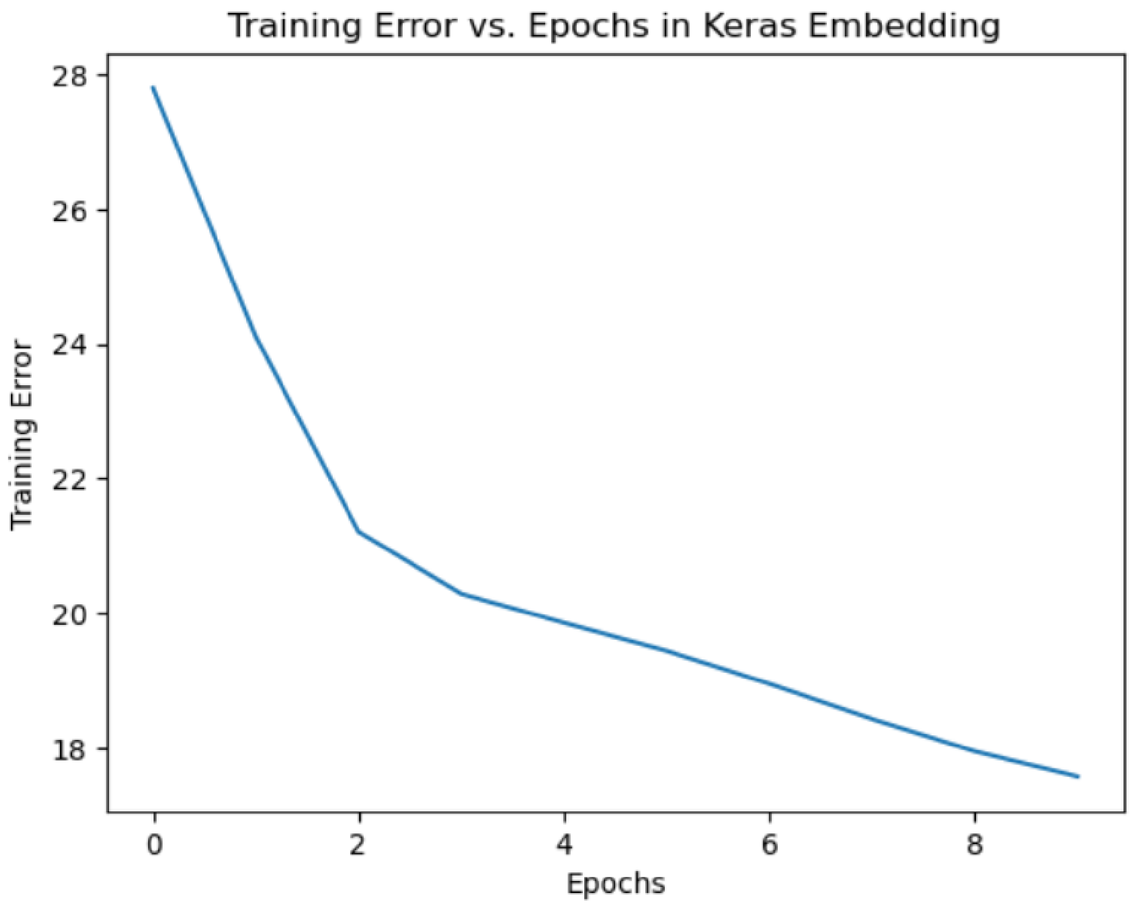

Figure 2: We track MSE as Epochs proceed.

the performance differences with differences in imputation approach.

2.8 DISCUSSION

The discussion of our results is certainly more difficult due to concessions we had to make due to data analysis. At first glance, it seems that the most effective algorithm is collaborative filtering. This should not surprise us, as collaborative filtering historically led to very good results and a wave of adoption of this technology. With A MSE of 14 at k latent variables, it is competitive with the system of Goldberg (2001), which had a mean normalized error of 0.187. The metrics used are somewhat different (MAE in the paper vs. MSE). Implementing MAE, we get a MAE of 1.57 with 10 latent factors which implies that in order for our approach to be better (20 years later), the deviation between max rating and min rating would have to be under 10 on a 20 scale. This is not the case, descriptive statistics situate it at around 19 (depending on sample, size of 200).

The hardest topic was that of nearest neighbours. Introducing more than 1 neighbour would give place to tradeoffs, since then in order to suggest a new recommendation, we would have to average the ratings of the best available jokes. Demanding that the same joke is present in every neighbour would be exponentially unfruitful with increasing number of neighbours. Incorporating jokes that were not rated by some neighbours but were highly rated by some others leads to uncertainty and a tradeoff in goals (do we prefer a higher confidence good joke or a joke that can be very good but in an uncertain way?). The evaluation here is hard to do, though it could be argued that we could use some representative users as a training set and then test our results on the rest. We did not commit to this route due to time constraints and the usage of different techniques such as clustering in order to implement this. For example, the centroids resulting from a k-means clustering algorithm could be used as representative users. It would be certainly interesting to further research on this and incorporate the tradeoffs into the algorithm. It should also be noted that restricting ourselves to representative users, we deal with the scalability problem of having to compute pairwise distances towards every existing user, for every new user.

Finally, regarding the neural network implementation, the end result was not quite efficient. The loss stabilized at around 15 MSE, higher than collaborative filtering. The theoretical side explaining the embeddings explains our procedure. Nevertheless, we should keep in mind that adding additional characteristics might help this algorithm (Murphy, 2022), while pure collaborative filtering cannot make use of additional characteristings. Thus, feature analysis of the content could help with the embeddings.

References

Goldberg, Ken et al. "Eigentaste: A Constant Time Collaborative Filtering Algorithm." Information Retrieval 4 (2001): 133-151.

S. Vidya Sagar Appaji, M. Kusuma Patnaik, M. Dileep Nagendra Kumar, K. Manoj Kumar, K. Satish; Movie recommendation system using machine learning. AIP Conf. Proc. 4 October 2023; 2794 (1): 020028.

Ricci, Francesco Rokach, Lior Shapira, Bracha. (2010). Recommender Systems Handbook. 10.1007/978-0-387-85820-3_1.

Giles, C. Bollacker, Kurt Lawrence, Steve. (2000). CiteSeer: An Automatic Citation Indexing System. Proceedings of 3rd ACM Conference on Digital Libraries. 10.1145/276675.276685.

Herlocker, Jon Konstan, Joseph Terveen, Loren Lui, John C.s Riedl, T.. (2004). Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems. 22. 5-53. 10.1145/963770.963772.

Gonçalves-Sá, J., Pinheiro, F. (2024). Societal Implications of Recommendation Systems: A Technical Perspective. In: Sousa Antunes, H., Freitas,

P.M., Oliveira, A.L., Martins Pereira, C., Vaz de Sequeira, E., Barreto Xavier, L. (eds) Multidisciplinary Perspectives on Artificial Intelligence and the Law. Law, Governance and Technology Series, vol 58. Springer, Cham. Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. J Big Data 9, 59 (2022).

Beel, J., Gipp, B., Langer, S. et al. Research-paper recommender systems: a literature survey. Int J Digit Libr 17, 305–338 (2016). Murphy, 2022. Probabilistic Machine Learning, The MIT Press

Vlachavas I., Kefalas P., Vasileiadis, N., Kokkoras F., Sakellariou I., (2020), Artificial Intelligence, University of Macedonia Press