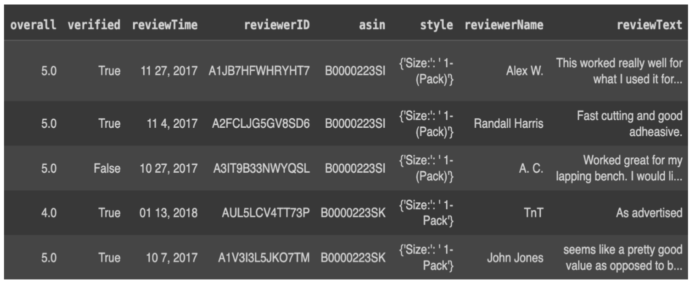
Time Weight Collaborative Filtering on Amazon Reviews

Dataset and Exploratory Analysis

The goal of this project is to predict ratings based on a dataset of Amazon reviews categorized as "Industrial and Scientific." This dataset was obtained from https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/#subsets. The data set contains ~77,000 product reviews. Each review in the JSON file consists of the ID of the reviewer, the ID of the product, the name of the reviewer, helpful votes of the review, the text of the review, rating of the product, time of the review (raw), time of the review (Unix time), as well as a couple of other features found on the dataset webpage (McAuley). This is a "smaller" subset of the dataset, intended for projects and experimentation as the data has been reduced to extract the 5-core, such that each of the remaining users and items has 5 reviews each. There are ~11,000 users and ~5300 products.



overall	verified	reviewTime	reviewerID	asin	style	reviewerName	reviewText
5.0	True	11 27, 2017	A1JB7HFWHRYHT7	B0000223SI	{'Size:': ' 1-(Pack)'}	Alex W.	This worked really well for what I used it for...
5.0	True	11 4, 2017	A2FCLJG5GV8SD6	B0000223SI	{'Size:': ' 1-(Pack)'}	Randall Harris	Fast cutting and good adheasive.
5.0	False	10 27, 2017	A3IT9B33NWYQSL	B0000223SI	{'Size:': ' 1-(Pack)'}	A. C.	Worked great for my lapping bench. I would li...
4.0	True	01 13, 2018	AUL5LCV4TT73P	B0000223SK	{'Size:': ' 1-Pack'}	TnT	As advertised
5.0	True	10 7, 2017	A1V3I3L5JKO7TM	B0000223SK	{'Size:': ' 1-Pack'}	John Jones	seems like a pretty good value as opposed to b...

Figure 1. A portion of each review in the dataset.

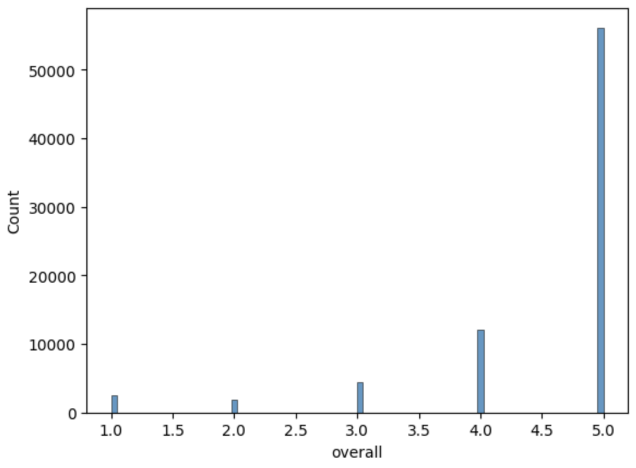


Figure 2. A histogram of the product ratings denoted by 'overall' in the dataset.

The dataset covers the period from April 27, 2003, to September 28, 2018. In order to gain a sense of how the ratings changed over time and to understand the temporal nature of the dataset better, I incorporated a sliding window. This technique involves maintaining a "moving average" using a window of fixed length.

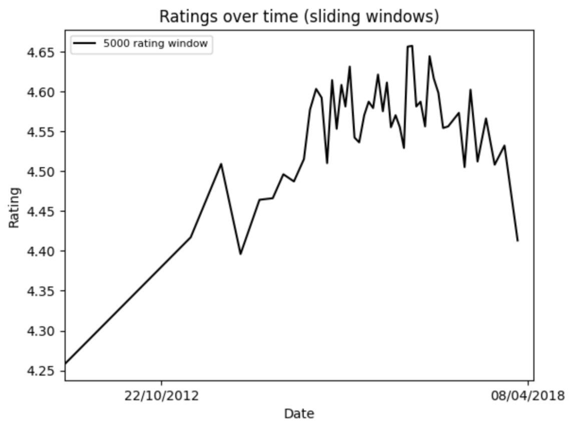


Figure 3. A sliding window of ratings and timestamps built with a dynamic programming approach.

To also acknowledge what a positive review consisted of, I created a word cloud to get a visual representation of the text data. Considering the products are categorized as "Industrial and Scientific," much of the terminology utilized in the reviews is very practical and straightforward (see Figure 4).

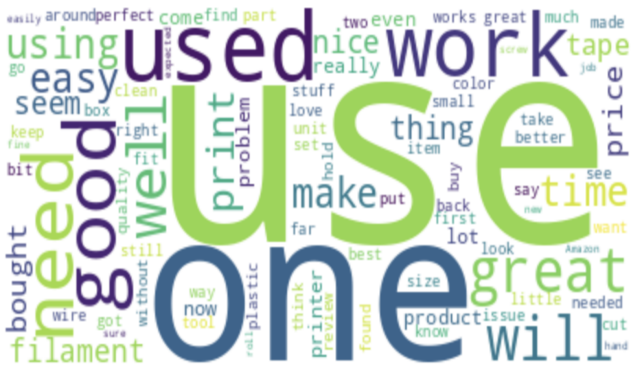


Figure 4. Word Cloud of the text in the product reviews generated using the wordcloud library in Python.

Predictive Task

I decided that the predictive task that was most relevant based on the exploratory analysis of the data was a rating prediction of what 'overall' from 1 to 5 stars a user would rate a product. To assess the validity of my model predictions I computed the mean-squared error

$$\text{MSE}: \frac{1}{N}||y - X\theta||_2^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - X_i \cdot \theta)^2$$

of the predictions and the true overalls in a test set. This requires a split of the data into a training set, used to optimize the model's parameters, and a test set, used to report how well I can expect the model to perform on unseen data. In order to avoid overfitting, the mean-squared error must be evaluated on data that was not used to train the model, i.e. the test set. Generalizing to new data is what constitutes an improved model. I selected the mean squared error to report performance in order to penalize larger errors as well as capture the variability better.

As a baseline model to serve as a reference to the actual model I will define in the following sections, I selected a random prediction algorithm. This predicts a random outcome as observed in the training data. My thought process behind this was that in the real world, data can not always be predictable. In scenarios such as this, a good baseline model would be a dummy classifier/regressor simply predicting random ratings. It allowed me to analyze whether the various machine-learning models were actually learning or not. This then made it intuitive to interpret which models were truly outperforming the baseline the most. Also, this baseline avoids using review data which is important because I need to make comparisons based on interaction-based models. The features I will use to construct these models are the reviewer's ID, the product's ID, ratings, and temporal features like the review date and the timestamp.

Model

After implementing several of the models covered in the latter half of class, I decided to select the time weight collaborative filtering model.

$$r(u,i) = \bar{R}_i + \frac{\sum_{j \in I_u \setminus \{i\}} (R_{u,j} - \bar{R}_j) \cdot \text{Sim}(i,j) \cdot f(t_{u,j})}{\sum_{j \in I_u \setminus \{i\}} \text{Sim}(i,j) \cdot f(t_{u,j})}$$

Figure 5. The formula for time-weight collaborative filtering. It can be derived by adapting a recommender that predicts ratings based on similarity functions through accounting for temporal factors.

This model is a form of temporal recommendation where interactions are weighted in terms of their recency. In Figure 5, $t_{u,j}$ is the timestamp associated with $R_{u,j}$, the rating. I defined the decay function to be based on exponential decay, and thus f(t) = $e^{-\lambda t}$, where lambda is a hyperparameter. I also utilized item-item Cosine similarity for Sim(i,j).

$$\text{Sim}(u,v) = \frac{\sum_{i \in I_u \cap I_v} R_{u,i} R_{v,i}}{\sqrt{\sum_{i \in I_u \cap I_v} R_{u,i}^2 \sum_{i \in I_u \cap I_v} R_{v,i}^2}}$$

Figure 6. The formula for Cosine similarity.

Regular collaborative filtering is a common item-centric recommendation technique that identifies the similarity between two items via a comparison of the current users' ratings of them. One caveat is that items rated at different timestamps are weighted equally. The time weight collaborative filtering algorithm takes this into account by finding time weights corresponding to items in the order that items are rated most recently. This allows those items that are rated more recently to contribute more to the predictions that the model makes.

The other models that I have considered for comparison are a factorization machine and an autoregressive model.

Factorization machines(FM) are a general purpose tool for modeling interactions amongst features, which for the dataset I selected are user history and the target item. This extends latent-factor recommender systems by incorporating arbitrary features. The idea is that every feature is associated with a low-dimensional vector.

$$f(x) = \underbrace{w_0 + \sum_{i=1}^{F} w_i x_i}_{\text{offset and bias terms}} + \underbrace{\sum_{i=1}^{F} \sum_{j=i+1}^{F} \langle \gamma_i, \gamma_j \rangle x_i x_j}_{\text{feature interactions}}$$

Figure 7. Factorization Machine Equation.

Autoregression is a temporal procedure in which given a time series, one can minimize the MSE by learning the weights of previous values. This model allows for the capture of periodic effects, e.g. the traffic of a website is most similar to its traffic 7 days ago.

To compare the four models being considered, the random baseline, autoregression, factorization machine, and time weight collaborative filtering, it would be helpful to consider both the advantages and disadvantages of each model. The dummy baseline model is advantageous because it is simple to implement and quick to compute while also serving as a benchmark to evaluate more complex models. However, the disadvantage is that it will yield lower accuracy than models that incorporate personalization. Autoregression is advantageous because it captures temporal patterns and dynamics in user behavior. The disadvantage of autoregression is that it could be susceptible to overfitting and requires a large amount of historical data. Factorization machines are advantageous because they are effective when dealing with sparse data that has high dimensions and also capture complex relationships between user/item features. The disadvantage of factorization machines is their scalability due to them being computationally intensive. The advantages of time weight collaborative filtering are that it considers the evolving user preferences as time passes and accounts for the temporal dynamics in the recommendation system. A potential drawback of time weight collaborative filtering is the choice of the time decay function, which heavily influences the model's performance and could be challenging to choose most optimally. I personally decided to select and optimize time weight collaborative filtering utilizing Cosine similarity and an exponential decay function because it seemed the most relevant in capturing the temporal and interactive aspect of the Amazon review data.

Related Literature

The Amazon "Industrial and Scientific" dataset I am analyzing comes from Professor Julian McAuley, UCSD, and Jianmo Ni, UCSD. It is a subset of a much larger dataset that contains product reviews and metadata from Amazon, including 233.1 million reviews from May 1996 to Oct 2018.

One key example of how the dataset was used was to test one-class collaborative filtering to model the visual evolution of fashion trends (He, McAuley). The fashion data is similar to the "Industrial and Scientific" subset of the Amazon data I am utilizing because they are both product reviews with several features. It was introduced in the paper "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," by Ruining He and Julian McAuley. The authors were interested in learning visual temporal dynamics from implicit feedback datasets (e.g. purchase histories of both clothing and accessories). The overall objective was to generate for each user in the dataset a "*time-dependent"* personalized ranking of those items that which they have not yet given feedback. The underlying method of their proposed model was matrix factorization, whose basic formulation they adopted. The conclusion was that visualization using trained models was useful in demonstrating the non-linear characteristics of the evolution of varying dimensions as well as showing how fashion had shifted over a 10-year period.

The method that I decided on, time weight collaborative filtering, was greatly inspired by the paper "Time Weight Collaborative Filtering," by Yi Ding and Xue Li, as well as the lectures and Personalized Machine Learning textbook by Professor Julian McAuley during CSE 158 at UCSD. After learning about similarity-based recommender systems based on computing similarities between two items/users using Jaccard, Cosine, Pearson, etc., I wanted to introduce a sort of temporal factor to this model, as Amazon themselves utilize an item-to-item collaborative filtering with Cosine similarity. I knew there was evidence backing the power of this already existing model, so I decided to look into adding a way to capture the temporal aspect of the Amazon product review dataset.

Introducing the temporal factor seemed valuable as even the winners of the famous Netflix Prize had to account for temporal dynamics. At the time in 2005, this novel algorithm was created by authors Yi Ding and Xue Li. They claimed that empirical studies would show that their algorithm substantially improved the precision of item-based collaborative filtering all while not introducing higher-order computational complexity. The core idea is to predict precisely a user's future purchase interest by deploying time weight. Their research concluded that their new algorithm definitely improved the precision of item-based collaborative filtering algorithms. In my own findings where I compared this model with two other models relevant to the predictive task, it seems that my conclusion was fairly similar.

Results

Model	Baseline (Random)	Autoregression	FM	Time Weight Collaborative Filtering
MSE	4.563	2.534	1.211	0.986

Figure 8. MSEs of each respective model being analyzed.

The time weight collaborative filtering model outperformed the other three models I analyzed, as the lower the MSE, the closer the forecast is to the actual rating (see Figure 8). The significance of the result is that considering the fact that Amazon utilizes Cosine similarity and collaborative filtering, it does seem intuitive that using a time weight collaborative filtering model with Cosine similarity would outperform the other observed models, as it provides even more information to work with. It emphasizes the importance of considering the temporal aspect of the dataset as it provides valuable insights into creating a more accurate model. It seems that for interpreting product reviews, the user and item features are also useful in computing the similarities between products to get a stronger model. Evidently, time weight collaborative filtering is a robust algorithm when it comes to constructing meaningful predictions in the context of product reviews in scenarios such as online shopping on websites like Amazon.com.

Works Cited

Justifying recommendations using distantly-labeled reviews and fined-grained aspects
Jianmo Ni, Jiacheng Li, Julian McAuley
Emprirical Methods in Natural Language Processing (EMNLP), 2019
pdf

Personalized Machine Learning
Julian McAuley

Cambridge University Press, 2022
pdf

Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering
R. He, J. McAuley
WWW, 2016
pdf

Image-based recommendations on styles and substitutes
J. McAuley, C. Targett, J. Shi, A. van den Hengel
SIGIR, 2015
pdf

Time Weight Collaborative Filtering
Yi Ding, Xue Li
2005
pdf

Amazon.com Recommendations Item-to-Item Collaborative Filtering
Greg Linden, Brent Smith, Jeremy York
Amazon.com
pdf