

---

# Amazon Product Recommendations Using Collaborative Filtering and Natural Language Processing

---

**Nikhil Aourpally**  
Nikhil.Aourpally@asu.edu

**Anthony Helmstetter**  
anthony.helmstetter@asu.edu

**Prajwal Paudyal**  
ppaudyal@asu.edu

**Anirudh Som**  
Anirudh.Som@asu.edu

## Abstract

In this project we will implement a Recommender System to provide suggestions based on Amazon user reviews across Amazon Instant Video, Music, and Book product categories. The Recommender System will be implemented using collaborative filtering based on product ratings. To increase recommendation accuracy, we will also perform natural language processing on user written reviews to better inform the recommender system of user preferences. We will assess the accuracy of the Recommender System with and without additional textual information contained in user reviews, and evaluate the sensitivity of our Recommender System by varying the number of reviews by user, and number of reviews per product.

## 1 Introduction

Modern consumers are inundated with a plethora of choices. Electronic retailers and content providers offer a huge selection of products, with unprecedented opportunities to meet users having a variety of special needs and tastes. Matching consumers with most appropriate products is key to enhancing user satisfaction and loyalty. For this reason, many online retailers have become interested in recommender systems, which analyze patterns of user interest in products. This allows them to provide personalized recommendations that suit each and every user's taste.

Personalized recommendation can add another dimension to a user's online shopping experience. Giants like Amazon.com, Netflix and other e-commerce leaders have made recommender systems a salient part of their websites. Recommender systems can be treated as a form of information filter. In general, a recommender system needs to complete two tasks [3]- First, we use a recommendation algorithm and the user's history logs to predict item ratings[4], which is not marked by the user before. Second [1], we generate a list of recommendation items for the user on the basis of the first task mentioned before, and also on a number of other factors like diversity, novelty, credits and so on. For every recommendation system, the first period is key and the core technology is the recommendation algorithm.

## 1.1 Collaborative Filtering

Recommender systems are based on one of two strategies [5], namely - content filtering and collaborative filtering (CF). The content filtering approach creates a profile for each user or product, in order to characterize its nature. These profiles allow programs to associate users with matching products. However, this strategy requires gathering external information that might not be available or easy to collect. On the other hand, collaborative filtering relies on past user behavior i.e. previous transactions or product ratings without requiring the creation of explicit profiles. Collaborative filtering analyzes relationships between users and interdependencies between products, to identify new user-item associations. The goal of collaborative filtering is to suggest the new items to users by predicting a score for an item (usually 0/1 or 1 – 5, etc) for a particular user based on his consume log of some old items as well as the log from other users with similar tastes.

Memory-based CF has two models - one is user-based and the other is item/product-based. The basic idea of user-based CF can be explained using the following example - suppose user A likes an item a, user B likes items a, b, c and user C likes items a and c. So user A, B and C are similar and item c is recommended to user A. Item-based CF calculates the similarity between items with all the history of the user's preference data, and thus the user is recommended with similar items. Model-based CF are developed using data mining, machine learning algorithms to find patterns based on training data which are used to make predictions for real data. Most of the recommender system algorithms have been derived from collaborative filtering. The biggest advantage of CF is that it is domain free. However, it suffers from cold start problems, due to its inability to address the system's new products and users. In this aspect, content filtering is superior.

## 1.2 Matrix Factorization

The primary areas of collaborative filtering [5] are the neighborhood methods and latent factor models. Neighborhood methods are centered on computing the relationships between items or, alternatively, between users. The item oriented approach evaluates a user's preference for an oriented item based on ratings of neighborhood items by the same user. A product's neighborhood are other products that tend to get similar ratings when rated by the same user. Latent factor models are an alternative approach that tried to explain the ratings by characterizing both items and users on, say 20 to 100 factors inferred from the ratings patterns. Most of the successful realizations of latent factor models are based on matrix factorization (MF).

The MF technique is gradually applied to personalized recommendation algorithm. It does a global optimization on object, so MF performs better than neighborhood methods in Root Mean Square Error (RMSE). In the basic form, MF characterizes both items and users by vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to a recommendation. One strength of MF is that it allows incorporation of additional information. When explicit feedback is not available, recommender systems can infer user preferences using implicit feedback, which indirectly reflects opinion by observing user behavior including purchase history, browsing history, search patterns, or even mouse movements. Implicit feedback usually denotes the presence or absence of an event, so it is typically represented by a densely filled matrix. However, ideally the MF approach will always prefer explicit feedback than implicit.

MF models map both users and items to a joint latent factor space of dimensionality  $f$ , such that user-item interactions are modeled as inner products in that space. Accordingly, each item  $i$  is associated with a vector  $q_i \in \mathbb{R}^f$ , and each user  $u$  is associated with a vector  $p_u \in \mathbb{R}^f$ . For a given item  $i$ , the elements of  $q_i$  measure the extent to which the item possesses those factors, positive or negative. For a given user  $u$ , the elements of  $p_u$  measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative. The resulting dot product,  $q_i^T \cdot p_u$ , captures the interaction between user  $u$  and item  $i$  - the user's overall interest in the item's characteristics. This approximates user  $u$ 's rating of item  $i$ , which is denoted by  $r_{ui}$ , leading to the estimate

## EQUATION 1

To learn the factor vectors ( $p_u$  and  $q_i$ ), the system minimizes the regularized squared error on the set of known ratings:

## EQUATION 2

Here,  $\kappa$  is the set of the  $(u, i)$  pairs for which  $r_{ui}$  is known (training set). The system learns the model by fitting the previously observed ratings. However, the goal is to generalize those previous ratings in a way that predicts future, unknown ratings. Thus, the system should avoid overfitting the observed data by regularizing the learned parameters, whose magnitudes are penalized. The constant  $\lambda$  controls the extent of regularization and is usually determined by cross-validation.

### 1.3 Alternating Least Squares

There are two approaches [5] to minimize equation2 stochastic gradient descent and alternating least squares (ALS). Both  $q_i$  and  $p_u$  are unknowns, and equation2 is not convex. By fixing one of the unknowns, the optimization problem becomes quadratic and can be solved optimally. Thus the ALS techniques rotate between fixing the  $q_i$ 's and fixing the  $p_u$ 's. When all  $p_u$ 's are fixed, the system recomputes the  $q_i$ 's by solving a least-squares problem, and next when all the  $q_i$ 's are fixed, the system recomputes the  $p_u$ 's by solving the least squares problem. This ensures that each step decreases equation2 until convergence.

In general, stochastic gradient descent is easier and much faster than ALS. However, ALS is more favorable than stochastic gradient descent in at least two cases ? First, when the system computes  $q_i$  and  $p_u$  independently of the other user factors respectively, the system can utilize massive parallelization of the algorithm. Next, when the system is centered on implicit data, because the training set cannot be considered sparse, and looping over each single training case as gradient descent does, would not be practical.

### 1.4 Natural language processing

Natural language processing (NLP) is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages. Modern NLP algorithms are based on machine learning, especially statistical machine learning. Prior implementations of language-processing tasks typically involve the direct hand coding of large sets of rules. Some of the tasks in NLP have direct real-world applications, while other more commonly serve as subtasks that are used to aid in solving larger tasks. Goal of NLP evaluation is to measure one or more qualities of an algorithm or a system, in order to determine whether (or to what extent) the system answers the goals of its designers, or meets the needs of its users.

## 2 Problem Description

The rapid growth of internet and e-commerce has brought the problem of information overload. Excessive information makes it difficult for a user to pick up what they really want and reduces the efficiency of information usage. Several search engines or e-commerce online platforms use personalized recommender systems in searching results. Personalized recommender systems not only performs better than the traditional ones but also adds another dimension to the user's experience, by analyzing patterns of the user's interest in products to suit their taste. Using Amazon product reviews, we implemented a recommender system to suggest new movies to Amazon users. The suggested movies are determined by highly rated products as rated by similar users. The similarity of users will be determined by the degree to which the users have rated identical products similarly. Our recommender system falls under the collaborative filtering category, in which we filter based on the similarity of users and not by the similarity of products.

To enhance the performance of our recommender system, we will also incorporate additional information derived from text reviews written by users using Natural Language Processing, to develop a more accurate metric of user similarity. We will evaluate the performance of our recommender system both with and without the incorporation of additional textual information using NLP and compare the results.

### 3 Methodology

#### 3.1 Base Recommender System

#### 3.2 Natural Language Processing

Most eCommerce retailers provide users an option to leave a review for products they have purchased. These reviews not only provide future users an overview of an item, but they also contain valuable information that the retailer can use. User reviews generally come with a text review. The data that we deal with comes with two sections of text reviews: the review summary and the text review (detailed). While the users provide a numerical review, which is what we base our main recommender system on, it is sufficiently clear that the review texts and summaries also contain some valuable information. We can leverage this information to give a finer grained score to each review.

Thinking intuitively, when a person looks at a movie to decide on whether or not to watch it, the recommender that is built on review scores gives some information but it misses out on the details that is contained in the text reviews. The goal behind using NLP in our system is so that we can somehow use this information in the review summary and text to fine-tune the review scores that go into the recommender system. For instance for two reviews that have the same initial rating of 4, one review summary and review text might be very positive, while the other one might be mediocre. We want a movie recommender system that is able to mine the information in the reviews and give a recommendation that is appreciative of this extra information.

While there are many recommendation systems that exist out there, a novelty of this recommendation system is that it finds a way to incorporate the sentiment information in the reviews that others don't take into consideration. A thing to be noted however is that although we use NLP to fine-tune the recommender, the results are not readily testable since we will ultimately compare against the numeric reviews in the test set, thus we don't expect the NLP system to do better than the base system in that respect. However we do fine tune our NLP system and the combination algorithms such that the overall test error is minimized.

**Stanford NLP.** The first method that we tried was the sentiment Analyzer that is provided by Stanford NLP lab [2]. This sentiment analyzer utilizes a 'Semantic Treebank' that includes fine grained sentiment labels for 215,154 phrases in the parse trees of 11, 855 sentences. This technique claims to increase the accuracy of predicting-fine grained sentiment labels for all phrases to 80.7 percent. This technique also successfully detects negation in a sentence which is very important for sentiment analysis on user reviews. After compilation of the NLP files, we get a 'pipeline' which takes a single sentence at a time and gives back a score that is between 0 and 4 which is convenient because we could scale this and combine it directly with our base recommender system. We built a program that would first get a sentiment value for the Review Summary section, then it would parse the Review Text section into individual sentences, get a sentiment value for each of those sections and then combine them and normalize it. We gave a slightly higher score to the Review Summary section than the review score section based on the formula:

```
java -cp "" -Xmx2g edu.stanford.nlp.pipeline.StanfordCoreNLP -annotators tok-  
enize,ssplit,pos,lemma,ner,parse,dcoref -file input.txt
```

$$(1.2 * ReviewSummaryScore + ReviewScore)/2.2$$

Although this approach seems to give us good results, we had to abandon it ultimately because it was slow and it could not give us results in a timely manner. The system took about 1 second to give a score back

for each sentence review. We were processing about 600,000 reviews, which roughly translates to 6 million sentences, which amounts to about 70 days of runtime. On top of that there was the overhead of combination and some isolated cases where the sentence spanned more than 60 words which exponentially increased the processing time.

**Python Blob.** The next approach we utilized for natural language processing was TextBlob [6]. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. Then we used the sentiment analysis module of this to receive a sentiment score between -1 and 1. The module returned a tuple of the form `Sentiment(polarity,subjectivity)`. The polarity score is a float within the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective. In did not utilize the objectivity score in our NLP processing primarily because we did not want to differentiate between a negative reviews that was very objective vs. one that was subjective. (Example. The movie was awful vs. I felt the movie was awful). However, after going through several movie reviews, we realized that a good portion of the reviews were plot descriptions and not actual sentiments on the movie. In this light, giving more weight to reviews with higher subjectivity might increase the effectiveness of the sentiment analyzer. Due to time constraints, we were not able to run a detailed test on this aspect and had to set this aside for future work.

As compared to the Stanford NLP processor, the textBlob algorithm was very fast. We used a similar method to parse the Review Summary vs. the Review Text and fed it to the textBlob. We also noticed that this NLP system was able to handle entire paragraphs at a time, which was one of the reasons it was much faster. This algorithm was able to compute the entire training set of close to 600,000 reviews in less than 29 minutes. The result of the analysis was a sentiment score of between -1 and 1. We used the following conversion method to convert this range to a range of 1 to 5 using the formula

$$Score(1 - 5) = 2 * score + 3$$

**Combination.**We noticed that the sentiment score that was returned was somewhat positively skewed and thus shifted the overall scores upwards instead of just providing a means to differentiate between reviews with the same initial score, thus we performed a simple mean shift to the NLP scores (shifted the mean to the integer scores). This made a lot of intuitive sense and also helped us reduce the testing error after we utilized the NLP system.

–insert diagrams for the mean shift.

We then used this shifted NLP score and combined based on the following formula.

$$BR + \alpha((AdNLP - BaseNLP)/4)$$

The  $\alpha$  here is the limiting factor which essentially limits how big of an adjustment that the NLP is allowed to have. For a value of 1, the NLP portion will be able to adjust the base score by at most 1 numeric score i.e. a review of 2 with the highest possible review score will be adjusted to a new score of 3. We performed a lot of tests on what would be the optimum value of the –alpha and settled on a value of 0.5. Since the reviews numeric score and the sentiment score didn't differ by such a huge degree in most cases, which meant that most of the adjustments stayed well below 0.5.

$$(1.2 * ReviewSummaryScore + ReviewScore)/2.2$$

## 4 Results

–Insert Results here –

## 5 Conclusions

–Insert conclusions here–

## References

- [1] Chuunhong Zhang, Bin Wang, Qing Liao. Weight based knn recommender system., 2013.
- [2] R. B. Bird, E. A. Smith, and D. W. Bird. The hunting handicap: costly signaling in human foraging strategies. *Behavioral Ecology and Sociobiology*, 50:9–19, 2001.
- [3] A. Tuzhilin, G. Adomavicius. Recommender systems handbook., 2010.
- [4] M. Reinders, J. Wang, A. Vries. A user-item relevance model for log-based collaborative filtering.
- [5] Chris Kolinsky, Yehudi Koren, Robert Bell. Techniques for recommender systems. *ATT*, page 2, 2009.
- [6] Chris Kolinsky, Yehudi Koren, Robert Bell. Techniques for recommender systems. *BTS*, 1:2, 2009.