Collaborative Filtering Recommendation Systems

A summary of the paper

Parth Patel - 04/05/2022

Reference - J. B. Schafer, D. Frankowski, J. Herlocker and S. Sen, "Collaborative Filtering Recommender Systems," The Adaptive Web, Lecture Notes in Computer Science, vol 4321. Springer, Berlin, Heidelberg. DOI:https://doi.org/10.1007/978-3-540-72079-9_9

Overview

The paper starts with overview of Collaborative Filtering (CF) systems by introducing terms *user, ratings* and *user-rating matrix* representing the ratings given by user to the items interacted. Then paper discuss why CF systems were introduced because of the need to finding quality items in a growing content base of items. Then the paper discuss about below topics

- The tasks for which users might use a CF system, things a CF system is good at, and the kinds of domains for which CF is appropriate
- Algorithms that CF systems employ
- How types of ratings in a CF system affect design choices
- How to evaluate and compare recommenders
- Open questions in the continuing development of CF systems

Use cases of CF Systems

Designers of web services should carefully identify the possible tasks users may wish to accomplish with their site as different tasks may require different design decisions. Below are few user tasks where CF systems are used

- Help me find new items I might like
- Advise me on a particular item
- Help me find a user (or some users) I might like
- Help our group find something new that we might like
- Help me with tasks that are specific to this domain

CF systems have 3 types of recommendation functionality

- Recommend: Show a list of items to a user, in order of how useful they might be. Often this is described as predicting what the user would rate the item, then ranking the items by this predicted rating
- Predict: Given a particular item, calculate its predicted rating
- Constrained recommendations: Given a particular set or a constraint that gives a set of items, recommend from within that set

Properties of Domains Suitable for CF

The paper list out few properties desired for a domain to make it suitable for CF. While these properties are not a must to apply CF, but the they provoke thought and discussion about what domains are easy or hard with collaborative filtering.

- Data Distribution - These properties are about the numbers and shape of the data.
    1. There are many items
    2. There are many ratings per item
    3. There are more users rating than items to be recommended
    4. Users rate multiple items

- Underlying Meaning - These properties are of the underlying meaning of the data
    1. For each user of the community, there are other users with common needs or tastes
    2. Item evaluation requires personal taste
    3. Items are homogenous

- Data Persistence - These are properties of how long the data is relevant
    1. Items persist
    2. Taste persists

Collaborative Filtering Algorithms

The paper classify CF algorithms in two categories: *non-probabilistic* and *probabilistic* algorithms. Algorithms are probabilistic if they are based on an underlying probabilistic model, they represent probability distributions when computing predicted ratings or recommendation lists.

Non-probabilistic algorithms - The popular CF algorithms are nearest neighbor algorithms. There are 2 classes of it: *user-based* and *item-based* nearest neighbor.

1. User-based algorithms - Algorithms generated predictions for users based on ratings from similar users called neighbors

2. Item-based algorithms - Algorithms generate predictions based on similarities between items. The prediction for an item should be based on a user's ratings for similar items

The paper then discuss the various formulas like by which the similarities are calculated.

Practical Challenges of User-Based Algorithms

- Ratings data is often sparse, and pairs of users with few co-ratings are prone to skewed correlations. For example, if users share only three co-rated items, it is not uncommon for the ratings to match almost exactly (a similarity score of 1). If such similarities are not adjusted, these skewed neighbors can dominate a user's neighborhood.

- Another problem with Pearson correlation is that it fails to incorporate agreement about a movie in the population as a whole. For instance, two users agreement about a universally loved movie is much less important than agreement for a controversial movie. Pearson correlation does not capture this distinction.

- Most importantly, calculating a user's perfect neighborhood is expensive, requiring comparison against all other users. Thus, in a naive implementation, the time and memory requirements of user-based algorithms scale linearly with the number of users and ratings. Researchers have tried techniques like subsampling and clustering to address this challenge.

Non-probabilistic Dimensionality Reduction Algorithms

Several algorithms reduce domain complexity by mapping the item space to a smaller number of underlying dimensions. Intuitively, these dimensions might represent the latent topics or tastes present in those items. The smaller latent dimensions reduce run-time performance needs and lead to larger numbers of co-rated dimensions. These techniques define a mapping between a user's ratings and their underlying tastes. An item’s prediction can then be generated based on a user’s underlying tastes

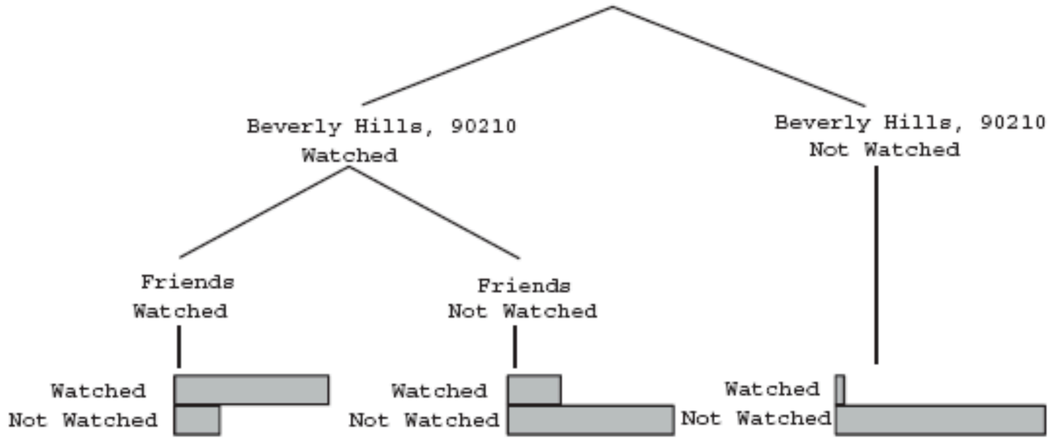
The primary challenge to utilizing such techniques is the mathematical complexity which can lead to challenges debugging and maintaining software utilizing those techniques.

Probabilistic Algorithms

Most probabilistic CF algorithms calculate the probability that, given a user *u* and a rated item *i*, the user assigned the item a rating of *r: p(r|u, i)*. We calculate a predicted rating based on either the most probable rating value or the expected value of *r*.

$$\mathbb{E}(r \mid u, i) = \sum_{r} r \,.\, p(r \mid u, i)$$

The most popular probabilistic framework involves Bayesian-network models that derive probabilistic dependencies among users or items. Some of the earliest probabilistic CF algorithms describe a method for deriving and applying Bayesian networks using decision trees to compactly represent probability tables. For example, fig shows that users who do not watch “Beverly Hills, 90210” are very likely to not watch “Melrose Place”. A separate tree is constructed for every recommendable item. The branch chosen at a node in the tree is dependent on the user's rating (or lack of rating) for a particular item. Nodes in the tree store a probability vector for user's ratings of the predicted item



Over-Arching Practical Concerns

Regardless of choice of algorithm, real-world CF systems need to address several problems that are generally not covered in research literature. Items and users with few ratings can inappropriately bias CF results. Below are the few approaches for rarely-rated entities

- *Discard rarely-rated entities*: Algorithms often only incorporate data with greater than k ratings. In a user-based algorithm, for example, we would discard neighbors with fewer than k co-ratings with the target user. Although this is a simple and clean approach it can decrease the coverage of the CF system

- *Adjust calculations for rarely-rated entities*: This technique adjusts calculations for rarely-rated entities by pulling them closer to an expected mean. For instance, Pearson similarities for users with few co-ratings may be adjusted closer to 0. CF systems often make the adjustment amount inversely proportional to the number of ratings. Although adjustment can be effective, tuning adjustment parameters can be difficult and unstable.

- *Incorporate a prior belief*: We can avoid skew by incorporating artificial data points that match an expected distribution. For example, we may believe that user's ratings will generally match a probability distribution p. We can incorporate this prior belief into user correlation calculation by including k artificial co-rated items whose ratings are independently drawn from p.

.

Acquiring Ratings: Design Tradeoffs

- Explicit ratings provided by users offer the most accurate description of a user's preference for an item with the least amount of data. However, because explicit ratings require additional work from the user, it can be challenging to collect ratings, particularly when creating a new CF service. On the other hand, implicit ratings observations of user behavior from which preference can be inferred are collected with little or no cost to the user, but ratings inference may be imprecise

- Another significant design decision involves choosing the explicit rating scale. The finer grained the scale, the more information you will have regarding each user's preference. Finer grained scales require more complex user interfaces

- The "***cold-start***" problem describes situations in which a recommender is unable to make meaningful recommendations due to an initial lack of ratings. This problem can significantly degrade CF performance. It can occur under three scenarios.

    1. *New User* - When a user first registers with a CF service, they have no ratings on record. Thus no personalized predictions can be given

    2. *New Item* - When a new item is added to a CF system, it has no ratings, so it will not be recommended

    3. *New Community* - The biggest cold-start problem is bootstrapping a new community. If a new service's value is in its personalized CF recommendations, then without ratings it may not have sufficient differentiating value – thus not retain users long enough to build up ratings

.

Evaluation

- Evaluation measures how well a collaborative filtering system is meeting its goals, either in absolute terms or in relation to alternative CF systems. There is no well-accepted metric that can evaluate all important criteria. Below are few points for evaluating CF systems

- Accuracy: Accuracy can either be measured as the magnitude of error between the predicted rating and the true rating, or the magnitude of error between the predicted ranking and the “true” ranking. Predictive accuracy is the ability of a collaborative filtering system to predict a user's rating for an item. The standard method for computing predictive accuracy is mean absolute error (MAE).

- Beyond Accuracy: While many of the published evaluations of CF systems measure accuracy, researchers and practitioners have come to learn that accuracy is not the only criteria of interest, and in some cases, may not even be the most important. Several other evaluation criteria have been explored.

    1. Novelty: It is the ability of a CF system to recommend items that the user was not already aware of.

    2. Coverage: It is the percentage of the items known to the CF system for which the CF system can generate predictions.

    3. Learning Rate: It measures how quickly the CF system becomes an effective predictor of taste as data begins to arrive.

    4. Confidence: It describes a CF system’s ability to evaluate the likely quality of its predictions.

    5. User satisfaction metrics: It is the metrics described above are only a sample of possible evaluation metrics.

Open Questions

This discusses some open questions in the field of collaborative filtering and they are grouped into algorithmic questions and temporal questions.

- Algorithmic questions:
    1. Evaluation metrics: For this many metrics of recommendation quality are proposed. Which ones capture what people perceive as good quality?
    2. Predicting well and recommending well at the same time. Efficient algorithms for recommendation may choose not to produce predicted values at all, or may choose to only store a small amount of information necessary to recommend some items. However, predicting a rating for a given user and item is an appealing application. Are there efficient, scalable algorithms that both recommend and predict well?
    3. Tagging: Social systems such as flickr, which allow users to tag things like photos with keywords, are increasing in popularity and have captured the imagination of many people. This is collaborative filtering system. There are many interesting research questions like How can collaborative filtering algorithms be applied to tags? Can tags be used in conjunction naturally with ratings?
- Temporal Questions: These questions are about the behavior of a collaborative filtering system over time.
    1. Item lifecycle
        a. When does an item have enough ratings to be accurately recommendable?
        b. When is an item a rising trend, falling trend, or a fad? Are many items like that?
    2. User lifecycle
        a. Can one identify the items for which it is possible to give good recommendations for a given user?
        b. How do old ratings affect a user’s recommendations, versus new ratings? Do user tastes shift over time? Can we detect it?
    3. Ratings database lifecycle
        a. When does a database have enough ratings to give good recommendations?
        b. Can one identify which items are likely recommendable?