Recommendation Engines

Bryan Turner

Nov, 2006

How does Collaborative Filtering differ from Statistical Inferencing?

Collaborative Filtering is a technique where a collection of items is rated by a collection of users. When a user wants a prediction for some un-rated item, the engine can use a “collaborative filtering” algorithm to generate a predicted rating for the item. The same algorithm can be used to rank a list of items and come up with a "top-10" recommendation.

The algorithm used to perform this task is very simple; find other rows or columns similar to the one being examined, and combine them to generate a rating. This technique produces ratings with a wide range of qualitative relevance. It is also generally difficult to compute the confidence level of the predicted ratings, owing to the generally poor statistical model used. In fact, the system is so bad, is difficult to see why it works at all.

Let's examine this in more detail; most movie recommendation systems (EachMovie, MovieLens, NetFlix, etc) have an extremely sparse item/user matrix. That is, each user has only rated a few of the thousands of movies, often 20 or less. The entire matrix may be missing up to 99% of the cells! From this matrix we need to generate recommendations for the missing cells.

When the collaborative filtering engine tries to find similar users by the movies they agree on, there is only a 1% chance that any two users have a movie rated in common! Let alone multiple movies or that the ratings themselves are similar (both users rated the movie 4 stars, etc). The algorithm attempts to fill out a 'neighborhood' of users who rated movies similarly to you. Then, it averages the ratings of the neighborhood for the movie of interest and reports that as the prediction.

So, let's take this by the numbers. Say 1 million people rate 10,000 movies at 99% sparseness (that's 100 ratings per person - 5 times more than the norm). Your neighborhood consists of the 1% of people whose movies intersect with yours at all (10,000 people). In a system with a 5-star scale, only 20% of them will have rated the intersecting movies close enough to be considered for prediction (2000 people). On average, only 1% of them will have rated the movie of interest (20 people). Finally, their ratings are averaged to return a prediction.

The wisdom of 20 people doesn't sound so bad until you examine how they relate to you. On average, each member of your neighborhood has only rated 1 movie in common with you! Does the fact that some random person

liked "Sleepless in Seattle" have any rational effect on giving you a prediction for "Fahrenheit 9/11"?

Therein lies the problem; there is no way for a collaborative filtering engine to understand why you liked or disliked a movie. Its prediction is similar to taking a very large digital picture with lots of missing pixels, and stepping back far enough until the image is a bit blurry, then trying to fill in the missing pixels. Not a very good technique by any stretch of the imagination.

So why do they work so well?

Here comes the human angle; we are not purely random processes. The 100 movies that we rate will tend to cluster around a few common movies (like "Sleepless in Seattle"), allowing the neighborhood for these movies to be much more dense. These popular items tend to act as focal points for the neighborhood selection algorithm.

Alas, does 1 million random people's ratings of "Sleepless in Seattle", "Titanic", "Braveheart", and "Pirates of the Caribbean" have any relevance to predict how you will enjoy "The Passion of Christ"? Not likely. This is another example of the typical popularity contests in school. What the popular people think is over-represented in the sample. But it explains why the basic technique works better than the numbers predict.

Semantic Modeling

For each movie (item, blog post, etc), there is a REASON you rate it. Maybe you like the actors, the cinematography, or it made you laugh. Perhaps it was a thought-provoking post or the item you purchased broke after the second use. That reason is important and should be captured by the rating.

In order to capture the reason, there has to be a place to put it. In a collaborative filter, there is only one 'slot' per item - the rating. There is no place to rate the acting or the special effects. In a model-based system, there are many different slots for each item, and you can enter ratings in the slot which most closely matches your intent.

Models can also capture static, factual information. For instance the actors and producer of a movie are factual and can be used to infer information about your rating. If you rated the acting high, then more weight should be given to other movies in which the same actors played a role, etc. Many facts exist which are not taken into account by collaborative filters; movie sequels, actors, genera, release dates, etc.

Models can capture non-factual information also; that some movie was "similar in spirit" to another, contained parodies, or any other type of relationship you can imagine. This information creates a link between the two items which is observable, but not necessarily provable. For instance, "The Matrix" and "Equilibrium".

Finally, models can capture meta-information which is not even related to the items themselves; are the user's ratings trustworthy, do you disagree with someone's rating, the user's demographics, age, gender, social network, etc.

Model-based techniques work much the same as collaborative filters: They search for clusters of information and infer the value of some missing data. However, they can produce much more interesting outputs than a basic collaborative filter:

- Given that a new movie is being made, with said actors and producer, in said genera, how many people are going to like it?
- What actor is the best lead-role in a comedy?
- What users are the best overall predictors of a movie's final rating? Perhaps they should be invited to a pre-screening..
- I want to watch a movie with my friends that none of us have seen before, which movie should we watch?

See the difference? Semantic models allow very deep and meta-level queries which are not captured by traditional collaborative filter engines.

Building a Semantic Model

Building the model requires a deep understanding of the system being modeled. All possible semantics must be accounted for, or the model will not be a good predictor. This leads to the idea of "sufficient statistics", a term describing all the parameters that a model must have in order to capture the semantics of the domain.

Coin tosses are a simple example. You don't need to capture the result of every coin toss, the ambient temperature, who tossed which coin, the date and time, etc. Just a total number of tosses, and the number of heads. These are the "sufficient statistics" for an ideal coin.

In large, complex systems like recommendation engines, it may not be possible to discover all the semantic elements in the system at the outset. It would be nice if we could update the model later, as we learn more about the system, but start with something now to get things rolling.

Bayesian Networks

Bayesian (or Belief) networks are a class of semantic models which capture a "prior" belief in a system model. For instance, we may create a movie model that has an overall rating, then ratings for acting and special effects. We specify our belief in the connections before collecting any ratings; perhaps we believe the acting influences the total rating by 80% , while special effects only influence it by 20%.

Next, we collect ratings. After some time, we can check the system model; given all the observations (some total ratings, some ratings for acting, and some ratings for special effects), is our prior belief that acting is 80% of the total correct?

We may find that neither acting nor special effects add up to the complete rating! If this is the case we can perform inferencing. That is, we infer from the data that there is some additional factor missing from the model. By creating a new, unlabeled, node and linking it into the model, we find this node influences the final score by 16%!

Someone, usually the model designer, labels this node to give it a semantic meaning. Let's say they predict that the node is "location". Now, we can collect observations on total rating, acting, special effects and location.

Later, we can again check for adherence to the model, and infer additional missing parts based on the observations. Eventually, we may end up with a model of 100s of nodes, some which are relevant and others which are not. A similar technique can be used to prune the irrelevant nodes from the network.

Ratings in Model-Based Systems

One interesting aspect of model-based statistical systems is that ratings no longer require a scale. A simple Yes/No is adequate in most cases. A rating scale may be synthesized from the aggregate yes/no ratings by using a statistical function called the Beta Function. This gives not only the rating (4-stars) but also the confidence level. A movie which is a confident 4-stars is often better than a movie which is a highly-speculative 5-stars. Think about it in terms of investments; a guaranteed 6% return may be much better than a risky junk bond with a 500% predicted return.

There is also a function that calculates the "expected value" of a risky financial decision, it uses Bayesian modeling just like the movie ratings, and it is just as applicable here. By applying the expected value function to each rating, we can order them into a top-N list. This mixes the confident winners with the risky up-and-comers in a single list.

The concept of expected values has progressed to modeling human behavior as well, and advanced models such as “cumulative prospect theory” achieve a better overall balance in the final ratings than pure expected value. As improvements are made to these theories, model-based systems will be the first to take advantage of them.

Gathering Ratings

From the user's perspective all this modeling is irrelevant. They want to find the answer to some question, and they don't want to be bothered by the ratings system. Luckily, model-based systems also improve the user's perceived value.

All statistical model-based systems can calculate "predictors"; empty rating slots which (if filled in by the user) will significantly improve the results of the system for that user. In movie-recommender terms, this means a movie like "Sleepless in Seattle" is not very useful as a predictor. Everyone has seen it, and most people like it. Knowing that you have seen or liked it does not tell the system much about you. But movies like "Fahrenheit 9/11" and "The Passion of Christ" can tell a great deal about the viewer in one stroke.

In order to quickly provide the most relevant information to the user, we need only calculate a small set of the best predictors, and ask the user to fill them in. Although there is a balance to be made; even if the predictor is "perfect" it may be a movie which is too obscure and thus no one has seen it (have you ever seen "Free Enterprise"?).

Collaborative Model Building

Finally, we have reached the theoretical/speculative part of this discussion. Given all the previous pieces, it is not difficult to imagine a massive system which automatically performs inferencing, pruning, and belief adjustment, and leverages the users to label new semantic nodes. The users would be able to rate not only the items, but also connections between items, other user's ratings of an item, etc.

The general process is quite simple, but the trick is presenting it to untrained users in a manner that they can grasp and utilize. To this end, it may be possible to present them with statements about an item and see if they agree with the statement or not. "This movie was enjoyable.", "I would call this an action movie", "The lead actor played his part well.", etc..

When new nodes are inferred, a user can be queried for a statement which fits the data; given a list of statements, enter one which expresses the unique property of this movie that these statements are describing. If the

statement is incorrect (or unintelligible), other users will disagree with it and it will eventually be pruned.

As an overhead process, a small group of moderators can elevate statements from the user-plane into the fact-plane, meta-plane, etc. This process allows the surviving memes to become focal points for future refinement.

References

For the skeptical, this paper also evaluated collaborative filtering vs bayesian networks and found bayesian techniques to out-perform collaborative filters overall, and do so with less memory and CPU.

http://citeseer.ist.psu.edu/breese98empirical.html

Pros:

- 41% improvement over cluster models in ranked scores (Top-N lists).
- Bayesian models were one-sixth the size of cluster models.
- Generated recommendations 4 times faster than cluster models.

Cons:

- Required 8 hours to calculate the model (300 nodes).