**Evaluation of Recommendation Systems**

There are several ways to evaluate a recommender system as we would see below. However, what to focus on would depend on cultural factors, what the business is trying to achieve. A very good way is A/B testing where controlled online experiments are conducted to see how users actually react to different components of the recommender system. We should always remember the **surrogate problem**. Accurate predicted ratings do not necessarily make a good recommender system.

**Accuracy Metrics**

* Mean Absolute Error (MAE)
* Root Mean Squared Error (RMSE)

However, in the real world, these metrics are not the most useful because most importantly people need to like what they were recommended.

**Evaluating Top-N Recommenders**

**Hit Rate**

* You generate top N recommendations for all of the users in your test set. If one of the recommendations in a user's top N recommendations is something they actually rated, you consider that a hit. Just add up all of the hits in your top-N recommendations for every user in your test set divide by the number of users, and that is your hit rate.
* Measuring Hit Rate is tricky, usually requires the used of Leave-One-Out cross validation. We compute the top N recommendations for each user in our training data, and intentionally remove one of those items from that users training data. Then we test our recommender system’s ability to recommend that item that was left out in the top N results it creates for that user in the testing phase.
* But it is a lot harder to get one specific item right while testing.

**Average Reciprocal Hit Rate (ARHR)**

* This metric is just like hit rate but accounts for where in the top N list you hits appear. So you end up getting more credit for successfully recommending an item in the top slot, than in the bottom slot.
* More user-focused metric, since users tend to focus on the beginning of lists.
* Example if we successfully predict a recommendation in slot 3, that only counts as 1/3. Whereas a hit in slot 1 receives the full weight of 1.0.

**Cumulative Hit Rank (cHR)**

* Throwing away hits if our predicted ratings fall below some threshold.
* Idea is not to get credit for recommending items to a user that we think they won’t actually enjoy.

**Rating Hit Rate (rHR)**

* Break it down by predicted rating score to get an idea of the distribution of how good the algorithm thinks recommended items are, that actually get a hit.

**Beyond Accuracy**

**Coverage**

* The percentage of possible recommendations (or user-item pairs) that your system is able to provide.
* May be inversely proportional to accuracy. If you enforce a higher quality threshold on the recommendations you make, then you might improve your accuracy at the expense of coverage.
* It can also give us a sense of how quickly new items in the catalogue will start to appear in recommendations.

**Diversity**

* Measure of how broad a variety of items the recommender system is putting in front of people. An example of low diversity would be a recommender system that just recommends the next books in a series that you’ve started reading, but doesn’t recommend books from different authors or movies related to what you’ve read.
* To measure diversity, we take the opposite of average similarity, so we subtract it from 1 to get a number associated with diversity. Average similarity would have been the average of some sort of computed similarity metric or score for every possible pair in the top-N list.
* Diversity isn’t always a good thing. Very high diversity can simply be achieved even by recommending completely random items, but those are not good recommendations by any means.
* Always need to look at diversity alongside metrics that measure the quality of the recommendations.

**Novelty**

* Measure of how popular the items are that we are recommending. What to do with it is in many ways subjective.
* Again, just recommending random stuff would yield very high novelty scores since the vast majority of items are not top sellers.
* Concept of user trust in a recommender system, people want to see at least a few familiar items in their recommendations that make them say “yeah, that’s a good recommendation for me. This system is good.”
* Need to strike a balance between familiar popular items and what we call serendipitous discovery of new items the user has never heard of before. The familiar items establish trust with the user and the new ones allow the user to discover entirely new things that they might love.

**Churn**

* Can measure how sensitive our recommender system is to new user behavior. If a user rates a new movie, does that substantially change their recommendations? If so, then the churn score will be high.
* Maybe just showing someone the same recommendations too many times is a bad idea. If a user keeps seeing the same recommendation but doesn’t click on it, at some point we should just stop trying to recommend it.
* Sometimes a little bit of randomization in the top-N recommendations can keep them looking fresh and expose users to more items than they would have seen otherwise.

**Responsiveness**

* Measure of how quickly the new user behavior influence our recommendations.
* More responsiveness would seem to be a good things but in the world of business we have to decide how responsive the recommender really needs to be.
* Recommender systems that have instantaneous responsiveness are complex, difficult to maintain, and expensive to build. We need to strike the balance between responsiveness and simplicity.