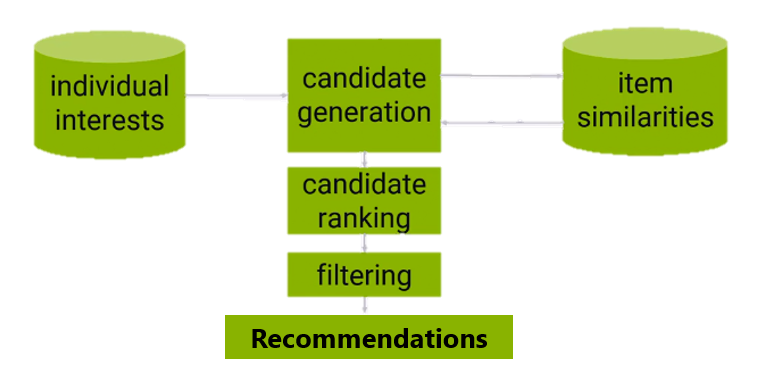
**Recommendation Systems Architectures**

**Top-N Recommenders**

The job of a Top-N Recommender is to produce a finite list of the best things to present to a given person. Success depends on the ability to find the best top recommendations for people so it makes sense to focus on finding things people will love and not the ability to predict the items people will hate.

Example architecture of item-based filtering:



**Data**

Conventionally, the data is stored in a NoSQL database like Cassandra, or MongoDB, or Memcached, where it has lots of data but with very simple queries.

Ideally, this interest data is normalized using techniques such as mean centering or z-scores to ensure that the data is comparable between users, but in the real world, you data is often too sparse to normalize it effectively.

**Candidates**

First step is to generate recommendation candidates, items we think might be interesting to the user based on their past behavior, so the candidate generation might take all of the items the user indicated interest in before and consult another data store of items that are similar to those items based on aggregate behavior.

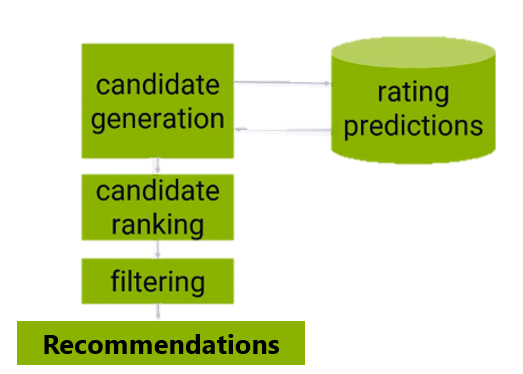
Example: Let's say you're making recommendations for me. You might consult my database of individual interests and see that I have liked Star Trek stuff in the past. Based on everyone else's behavior, I know that people who like Star Trek also like Star Wars, so based on my interest in Star Trek, I might get some recommendation candidates that include Star Wars stuff.

Then there is also **candidate ranking**. Many candidates will appear more than once and need to be combined together in some way, maybe boosting their score in the process, since they keep coming up repeatedly. After that, it can just be a matter of sorting the resulting recommendation candidates by score to get our first cut at a top-N list of recommendations.

**Filtering**

Some filtering will be required before presenting the final sorted list of recommendation candidates to the user. This filtering stage is where we might eliminate recommendations for items the user has already rated, since we don't want to recommend things the user has already seen. We might also apply a stop list here to remove items that are potentially offensive to the user or remove items that are below some minimum quality score or minimum rating threshold. It's also where we apply the N in top-N recommenders and cut things off if we have more results than we need.

Example of architecture with rating predictions for every user:

****

The candidate generation phase is then just retrieving all of the rating predictions for a given user for every item, and ranking is just a matter of sorting them. This requires you to look at every single item in your catalogue for every single user, however, which isn’t very efficient at runtime.