Lecture 5: Collaborative Filtering II

Derek Bridge

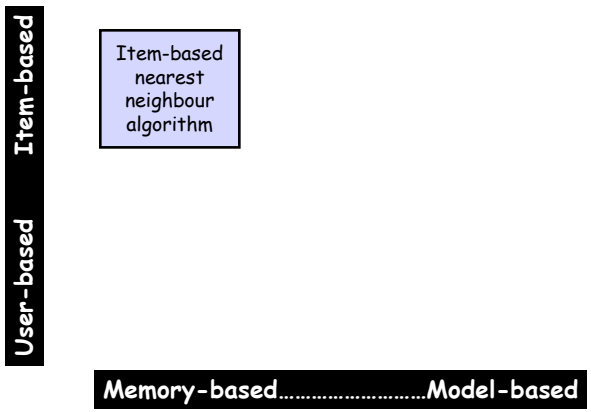
Challenges

- Improving time and space efficiency
- Handling cold-start issues
- Coping with sparsity
- Enhancing security & privacy
- Detecting and overcoming attacks

Improving time and space efficiency

- The user-based nearest-neighbours algorithm compares the active user to potentially every other user
  - there may be tens of millions of other users
  - in some domains, the number of items can be very large too (e.g. web pages)
  - finding the neighbours takes time and space that grows linearly with the number of users and the number of items
- This has led academics and practitioners to investigate item-based approaches

Types of CF algorithm



Ratings matrix

- Observation: we can compute similarity between columns (item-item similarity) just as easily as we can compute similarity between rows (user-user similarity)

	Alien	Brazil	Crash	Dumbo	E.T.	Fargo
Ben		2	5	3	1	2
Col	5	5		3	4	
Deb					3	
Edd	5	4	2	4	3	3
Flo	5	5	4			
Ann	2			4	3	5

Item-based CF prediction algorithm

- Intuition: predict the rating based on this user's ratings for other items and how similar those other items are to *i*
- To predict a rating for active user *a* and item *i*:
  - For every item *j* that user *a* has rated,
    - Use profiles of all users who have rated both *i* and *j* to compute the similarity between *i* and *j*, *sim(i, j)*
  - Let *NN* be *i*'s nearest neighbours, i.e. the set of size *k* for which *sim(i, j)* is highest
  - Compute a predicted rating, *pred(a, i)*, from *NN*'s ratings
- Worked example
  - We'll predict Ann's rating for Brazil
  - What similarities will we compute?

Similarity, *sim(i, j)*

- Many systems use an *adjusted cosine correlation*

$$sim(i,j) = \frac{\sum_{u \in RB_{i,j}} (r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in RB_{i,j}} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{u \in RB_{i,j}} (r_{u,j} - \bar{r}_u)^2}}$$

- It is computed over all users *u* who have rated both *i* and *j*, $RB_{i,j}$
- $\bar{r}_u$ is the average over all of user *u*'s ratings

Example: *sim(Brazil, Alien)*

	Alien	Brazil
Ben		2
Col	5	5
Deb		
Edd	5	4
Flo	5	5
Ann	2	

$$sim(i,j) = \frac{\sum_{u \in RB_{i,j}} (r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in RB_{i,j}} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{u \in RB_{i,j}} (r_{u,j} - \bar{r}_u)^2}}$$

Example

- The similarities
  - sim(Brazil, Alien) =
  - sim(Brazil, Dumbo) = -0.61
  - sim(Brazil, E.T.) = 0.28
  - sim(Brazil, Fargo) = 0.18
- Suppose *k = 3*
- Brazil's 3 nearest neighbours are Alien, E.T., Fargo

Predicted rating, *pred(a, i)*

- Following the same reasoning as before, we take a weighted average:

$$pred(a,i) = \frac{\sum_{j \in NN} r_{a,j} \times sim(i,j)}{\sum_{j \in NN} sim(i,j)}$$

- (This time *NN* is not a set of most similar users who have rated *i*; it is a set of most similar items rated by *a*)
- There is no need to adjust here for users' mean ratings because the only ratings used here come from the same user

Example

	Alien	E.T.	Fargo
Ann	2	3	5

$$pred(a,i) = \frac{\sum_{j \in NN} r_{a,j} \times sim(i,j)}{\sum_{j \in NN} sim(i,j)}$$

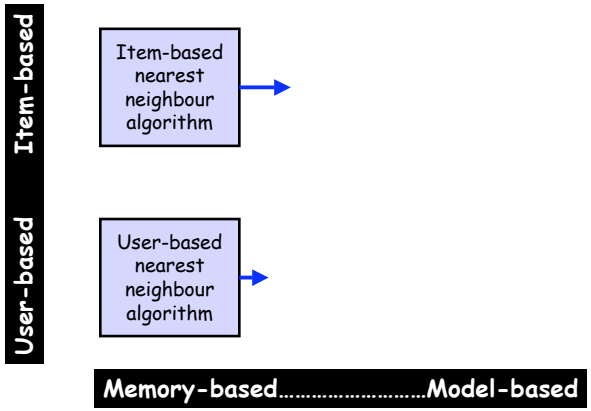
Item-based CF recommendation

- To recommend items to active user *a*:
  - you can take the approaches we saw before, e.g. computing predictions
  - Amazon simply finds items that are similar to ones you liked (viewed, added to wish list, purchased,…) and recommends these items

User-based versus item-based

- User-based seems 'truer' to word-of-mouth recommendations
- They have comparable accuracy
  - (the most accurate approach changes all the time)
- Time and space efficiency
  - in item-based prediction, you are only computing the similarity between *i* and each item that *a* has rated
- It is claimed that item-item similarity is likely to be more stable than user-user similarity
  - hence, item-based is more amenable to precomputation

Types of CF algorithm



Precomputation

- User-based
  - precompute all user-user similarities
  - find neighbours on-line
  - make predictions/recommendations on-line
  - periodically, recompute user-user similarities
- Item-based
  - precompute all item-item similarities
  - precompute each item's *k* nearest neighbours
  - make predictions/recommendations on-line
  - periodically, recompute item-item similarities & neighbours
    - but less often than you would for user-based

Handling cold-start issues

- An initial lack of ratings
- New user:
  - A new user has no ratings, so we cannot form neighbourhoods (except randomly)
  - Solutions
    - ask the user to rate some initial items (which items?)
    - display non-personalized predictions, e.g. based on population averages
    - use a hybrid recommender
- New item:
  - A new item has no ratings so we will not recommend it
  - Solution
    - use a hybrid recommender

Cold-start issues

- New community:
  - Catch-22
    - without ratings we can't recommend well,
    - but until we recommend well, people won't hang around and give us ratings
  - Solutions
    - incentivize early adopters
    - have some other service on offer at the same time
    - take ratings data from another source
    - use a hybrid recommender

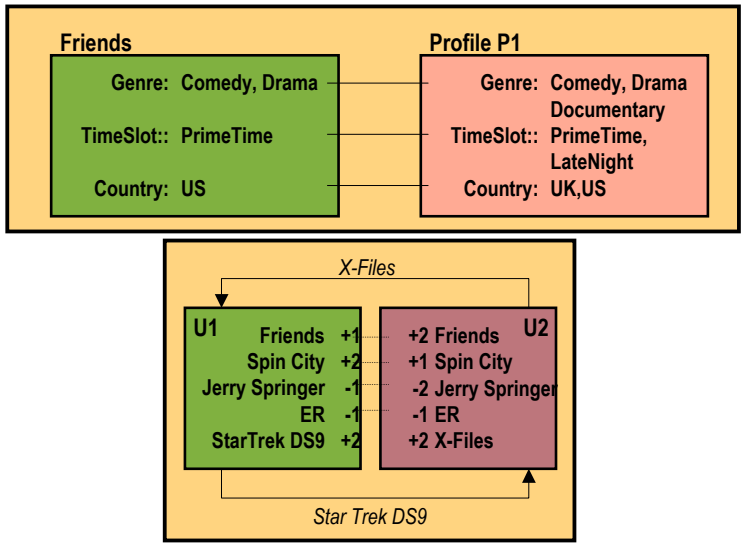
Hybrid systems

- *A hybrid* recommender system combines multiple techniques to achieve some synergy
- In particular, where complementary, one technique can compensate for another's weaknesses, e.g.
  - while a CF system is experiencing cold-start problems, recommendations to the new user or for the new item can still be made by some kinds of CB system
  - once rating profiles are of adequate size, the CF can make the serendipitous recommendations that the CB system cannot
  - grey sheep (users who do not consistently agree/disagree with others) can receive CB recommendations

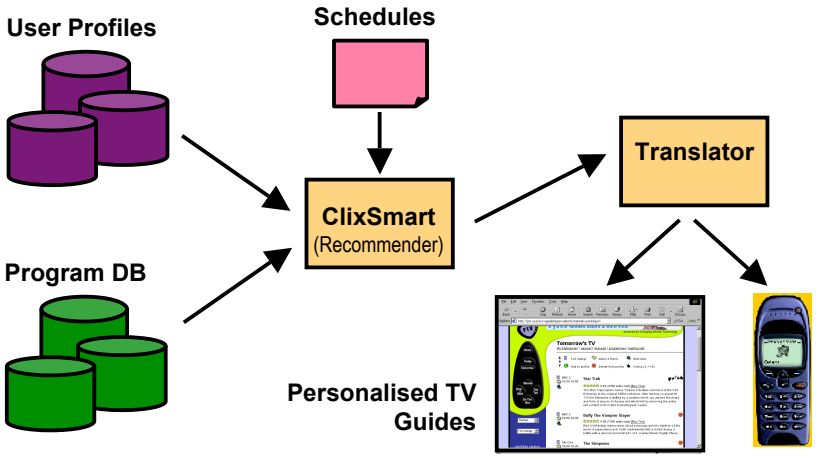
A hybrid system: PTV

- Problem
  - hundreds of TV channels, thousands even
  - in USA, *"TV Guide"* is 400+ pages
- Solution:
  - personalized TV guide
- Launched 1999
  - E.g. ran as MyTv on the Irish Times website
  - Acquired 20,000 users (3-5% of the Irish Internet population of the time)
  - But later discontinued

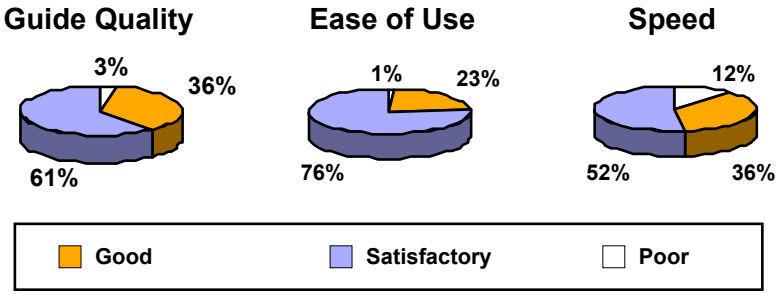
PTV: CB and CF



PTV



PTV: user trials (6 months)



Coping with sparsity

- Consequences for user-based CF
  - may not be able to find enough users who have rated *i*
  - users who have rated *i* may have no or few items in common with *a*
  - may find fewer than *k* neighbours (esp. if we exclude those with negative similarity)
  - may have to make do with neighbours with rather low similarity to *a*
  - …
- Similar issues for item-based CF
- Solution: inferred ratings(?) or hybrid systems

Enhancing security & privacy

- Concerns about security & privacy of data stored in centralized repositories
- One response is investigation of distributed peer-to-peer CF systems
  - maybe this increases privacy issues!
- Another response is investigation of ways of not revealing actual ratings
  - Encryption
  - Perturbation
  - Obfuscation
  - Adding imprecision
- Given auxiliary info, the identity of some users can also be revealed
  - people who connect clusters of different tastes are very susceptible

Detecting & overcoming attacks

- CF systems are vulnerable to attack
  - *profile injection attacks*
- *Shilling* attacks
  - *Nuke attacks:* inject profiles that give low ratings to competitors' items
  - *Push attacks:* inject profiles that give high ratings to your own profiles
- Other attacks
  - Inject random profiles to create mayhem
  - Inject profiles for 'social vandalism'
    - E.g. "Amazon blushes over sex link gaffe"
- Some believe user-based algorithms are more susceptible

Desirable extensions

- Reporting prediction and recommendation confidence
- Explaining recommendations, and enhancing scrutability
- Incorporating explicit notions of user trust
- Making recommendations that are sensitive
  - to short-term, ephemeral goals
  - to time and place, etc.
- Recommending bundles of items
- Recommending to groups of users
- Recommending across domains