CS918: LECTURE 16

Recommender Systems

Arkaitz Zubiaga, 26th November, 2018

LECTURE 16: CONTENTS

- What is a recommender system?
    - Applications.
- Types of recommended systems.
    - Collaborative Filtering.
    - Content-based Recommender Systems.
    - Hybrid Recommender Systems.
- Evaluation.

WHAT IS A RECOMMENDER SYSTEM?

- **Recommender system** (short **RecSys**): **information filtering** system, whose aim is to **predict items that a user will like**.
  - As opposed to IR, user does not specify an information need (e.g. a query).
  - Instead, we look at a user's features (e.g. history of items they liked in the past) to predict items they might like.

RECOMMENDER SYSTEMS: EXAMPLES

WARWICK

- User has eaten in **Chinese and Thai restaurants** in the last 3 months; can we recommend them **another restaurant**?

- User went to London for **rock concert** last week; can we recommend them **another future event**?

DO USERS FOLLOW RECOMMENDATIONS?

- Numbers suggest they do!
  - **Netflix:** 2/3 of the movies watched are recommended.
  - **Google News:** recommendations generate 38% more clickthrough.
  - **Amazon:** 35% sales from recommendations.

    ”users who bought X also bought Y and Z”

TACKLING THE RECOMMENDATION TASK

WARWICK

- Build model that automatically predicts if a user will like an item (e.g. whether they would give it a 5-star rating).
- Features that can be used:
  - Past behaviour.
  - Relations to other users.
  - Item similarity.
  - Context.
  - etc.

SERENDIPITY

- Users do tend to like **serendipity** (unsought finding)!
  - **Don't recommend the obvious:**
    items they **already know** or would have found anyway.
  - Expand the user's taste into **neighbouring areas**.

- e.g. user **has been to Chinese and Thai restaurants. Recommend Vietnamese restaurant!** How?
  - While different, they have overlapping features.
  - Others who go to Chinese/Thai, also go to Vietnamese.

TYPES OF RECOMMENDER SYSTEMS

- **Collaborative Filtering:** recommend based on past behaviour.
- **Content-Based:** recommend based on features extracted from item contents.
- **Other approaches:** Personalised Learning to Rank, Demographic, Social recommendations.
- **Hybrid approaches:** combinations of any of the above.

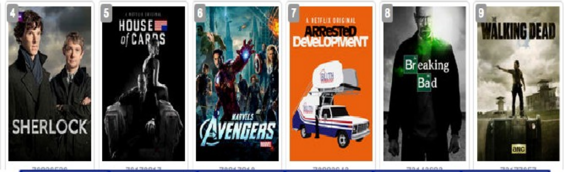
COLLABORATIVE FILTERING RECOMMENDER SYSTEMS

COLLABORATIVE FILTERING

- We have M users and N items in our system.
  - We can build a matrix.

- Each user has viewed, liked, rated,… some of the items.
  - e.g. 1 if viewed, 0 otherwise.
  - or better, 1-5 stars if they rated it, 0 otherwise.

COLLABORATIVE FILTERING: EXAMPLE MATRIX



2		2	4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

COLLABORATIVE FILTERING: STEPS

1. **Retrieve** set of **items D the user U liked**.
2. Find **users** with **similar items liked**.
3. **Retrieve** sets of **items** $\{L_1..L_u\}$ those **similar users liked**.
4. **Make predictions** on items in $\{L_1..L_u\}$ that are not in D based on how much user U will like them.
5. **Recommend top N items** to user U based on above predictions.

COLLABORATIVE FILTERING: TYPES

WARWICK

- **Two types of Collaborative Filtering (CF) systems:**
  - **Memory-based Collaborative Filtering.**
    Look at patterns in past behaviour to recommend new items.
    We'll focus on this one.
  - **Model-based Collaborative Filtering.**
    Using machine learning models to make predictions.

MEMORY-BASED COLLABORATIVE FILTERING

WARWICK

- **Memory-based CF** can be of **two types:**
    - **User-based Collaborative Filtering.**
      What can I recommend to user U based on what other similar users liked?
    - **Item-based Collaborative Filtering.**
      What can I recommend to user U that is similar to the items that user U liked?

USER-BASED COLLABORATIVE FILTERING

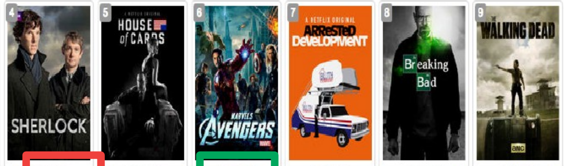
2		2	4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

They seem to have similar tastes, i.e. liked and disliked the same items

We want to recommend something to this user

We can recommend this item to the user.

ITEM-BASED COLLABORATIVE FILTERING



2		2	4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

He really liked this item.

We want to recommend something to this user

He'll like this one as it has similar ratings (for the 2 cases where we have ratings for both)

COLLABORATIVE FILTERING

- We can use **different metrics to compute similarity** between users or items, usually:
  - Cosine similarity.
  - Pearson correlation coefficient.

- NOTE: **similarity between sets of ratings** by two users or for two items. We're not using the content for anything!

USING COSINE SIMILARITY FOR CF

- Cosine similarity between two users u and u'.

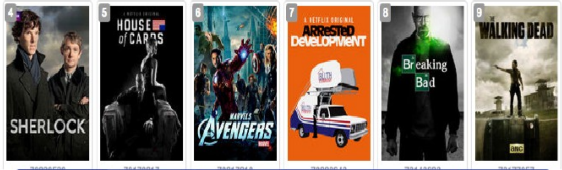
$$sim(u,u') = cos(\theta) = \frac{\mathbf{r}_u \cdot \mathbf{r}_{u'}}{||\mathbf{r}_u|| \, ||\mathbf{r}_{u'}||} = \sum_i \frac{r_{ui} r_{u'i}}{\sqrt{\sum_i r_{ui}^2} \sqrt{\sum_i r_{u'i}^2}}$$

- We can then predict new ratings for u.

$$\hat{r}_{ui} = \frac{\sum_{u'} sim(u,u') r_{u'i}}{\sum_{u'} |sim(u,u')|}$$

Sum of similarity-weighted ratings by others, normalised by the number of others we're taking into account.

COSINE-BASED RATING PREDICTION



2		2	4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

Target user

$$sim(u, u') = \sum_i \frac{r_{ui} r_{u'i}}{\sqrt{\sum_i r_{ui}^2}\sqrt{\sum_i r_{u'i}^2}}$$

A = (5*0 + 4*4 + 1*2) = 18
B = $\sqrt{}(5^2+4^2+1^2)$ * $\sqrt{}(4^2+2^2)$ = 28.98

sim(u, u') = A / B = 0.62

COSINE-BASED RATING PREDICTION

2		2	4	5		0.26
5		4			1	0.62
		5		2		0.83
	1		5		4	0.28
		4			2	
4	5		1			0

Target user

COSINE-BASED RATING PREDICTION

	2		2	4	5		0.26
	5		4			1	0.62
			5		2		0.83
		1		5		4	0.28
Target user			4			2	
	4	5		1			0

$$\hat{r}_{ui} = \frac{\sum_{u'} sim(u,u') r_{u'i}}{\sum_{u'} |sim(u,u')|}$$

r_{ui} = (5 * 0.26 + 2 * 0.83) / 2 = 1.48

COSINE-BASED RATING PREDICTION

WARWICK

2		2	4	5	
5		4			1
		5		2	
	1		5		4
1.21	0.14	4	0.81	1.48	2
4	5		1		

Target user

Two candidates for recommendation, moves 1 and 5.

Alternatively, we could've been more restrictive, e.g. only consider very similar users for rating prediction, like u_2.

COLLABORATIVE FILTERING: PROS AND CONS

- **Pros:**
  - **Easy to implement:** users and items can be treated as IDs, no content is processed.
  - **Performs reasonably well** when we have a history of likes.
- **Cons:**
  - **Cold start** and **popularity bias.**
  - **Can't perform well** when we lack sufficient history.
  - **Context is ignored** (e.g. if I move to a different city, don't recommend me restaurants in the previous city).

LIMITATIONS OF COLLABORATIVE FILTERING

- **Cold start:**
  - There needs to be enough other users already in the system to find a match.
    - What if the user base if so small that there isn't any other users that like the same items as I do?
  - Likewise, new items need to get enough ratings.
    - How do we know who will like a new movie released today if nobody has rated it?

LIMITATIONS OF COLLABORATIVE FILTERING

WARWICK

- **Popularity bias:**
  - We can end up having a tendency to recommend items that everybody likes.
  - Items from the tail only rated by a few will never be recommended.
  - Indeed, we can end up recommending the same stuff over and over… boring!

CONTENT-BASED RECOMMENDED SYSTEMS

CONTENT-BASED RECOMMENDATIONS

- Recommendations **based on the content** of items rather than on other users' opinions/interactions.

- Extract e.g. linguistic **features from items** the user viewed in the past.
- Build **machine learning model** that will find similar items that the user will like.

CONTENT-BASED RECOMMENDATIONS

- NOTE: we're completely **ignoring other users' preferences** and tastes.

- We're making recommendations for a user based **solely** on the profile built up by **analysing the content of items** that user has rated in the past.

CONTENT-BASED RECOMMENDATIONS

WARWICK

- The **three components** of content-based recommenders:
  - **Preprocessing** and feature extraction.
  - **Offline training** from labelled data (e.g. ratings).
  - **Online generation** of recommendations.
    - Here we **need to be efficient** to provide (near) real-time recommendations → many techniques we've learned for information retrieval are applicable.

CONTENT-BASED RECOMMENDATIONS

WARWICK

- **Preprocessing:**
  - **Stop-word removal:** remove very common words (aka high-IDF words) such as “a”, “the”, “of”, “from”.
  - **Stemming:** to maximise overlap of keywords.
  - **Phrase extraction:** identify common n-grams for the domain at hand (e.g. movies).

CONTENT-BASED RECOMMENDATIONS

WARWICK

- Types of features we can use:
  - **Metadata:** movies with same director, music of same genre, restaurants with similar cuisine, etc.
  - **NLP over content:** overlapping named entities across movies, similar lyrics for music, restaurants with similar menus, etc.

  As in information retrieval, **TF-IDF can be useful here too** → rare words are more informative!

CONTENT-BASED RECOMMENDATIONS

- Types of features we can use:
  - **Opinion mining:**
    process a user’s reviews, can we identify types of items they like/dislike from their reviews? e.g.

    I like this horror movie as much as I hate romantic movies!

    Even if we don’t have ratings of the user on romantic movies, can we leverage this review to never recommend them romantic movies?

CONTENT-BASED RECOMMENDATIONS

- Types of features we can use:
  - Of course.. **word embeddings**!
    - As with many other NLP tasks, they can be useful to identify semantic similarities between items.
  - And more… **be creative**!

CONTENT-BASED RECSYS: PROS AND CONS

WARWICK

- **Pros:**
  - No need for data on other users. **No cold-start** problem.
  - Able to recommend to **users with unique tastes**.
  - Able to recommend **new and unpopular items**. No first-rater problem.
  - We can provide better explanations of the reasons behind a recommendation, i.e. the **features that the recommended item has in common** with what they've viewed before.

CONTENT-BASED RECSYS: PROS AND CONS

WARWICK

- **Cons:**
  - Need to **extract features** from content, more difficult for e.g. video.
  - **Difficult** to implement **serendipity**.
  - **Easy to overfit** (e.g. for a user with few data points we may “pigeonhole” them).

HYBRID RECOMMENDER SYSTEMS

HYBRID APPROACHES

- Combine content-based methods and collaborative filtering methods to get the best of both.
  - **Content-based methods** will help overcome the cold-start problem as well as to make recommendations when there are no similar users.
  - **Collaborative filtering methods** will avoid pigeonholing the user with recommendations that are very similar in content, with no novel stuff.

HYBRID APPROACHES

- We can also combine other methods:
  - **Social recommendations:** what do my friends/followers like?
  - **Demographics:** what do people of my age or from my city like?
  - **Serendipity:** include something new, from the long tail of unpopular items, the user might want to give it a go!

EVALUATION

EVALUATION

- **Evaluation is tricky!**
  - We don’t know if a user WILL LIKE something.
  - We only know what they DID LIKE.

EVALUATION

WARWICK

- If we are lucky enough to have a **platform with a good user base.**
  - You can try your recommendation algorithm with a few users.
  - Do **A/B test**.
    - Group A will see our recommendations.
    - Group B will see something else, e.g. another recommendation algorithm.
  - Which group clicks more on the recommended items, A or B?

EVALUATION

- However, if you don’t have a platform with a good user base… (as in most of the cases)

  we need a **dataset for evaluation**.

- We can **collect a dataset**, but we will only have data for items the user have **already rated**.

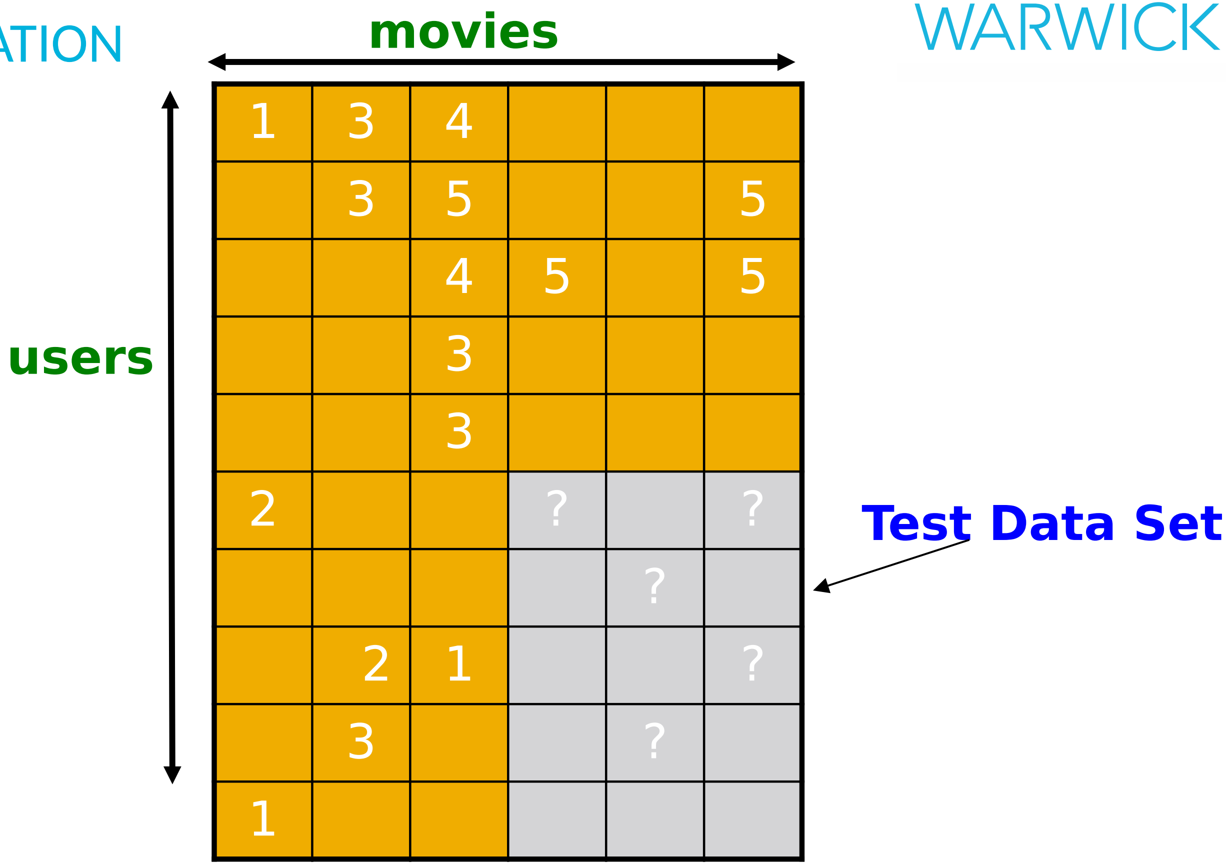
DATASET FOR EVALUATION

movies

users

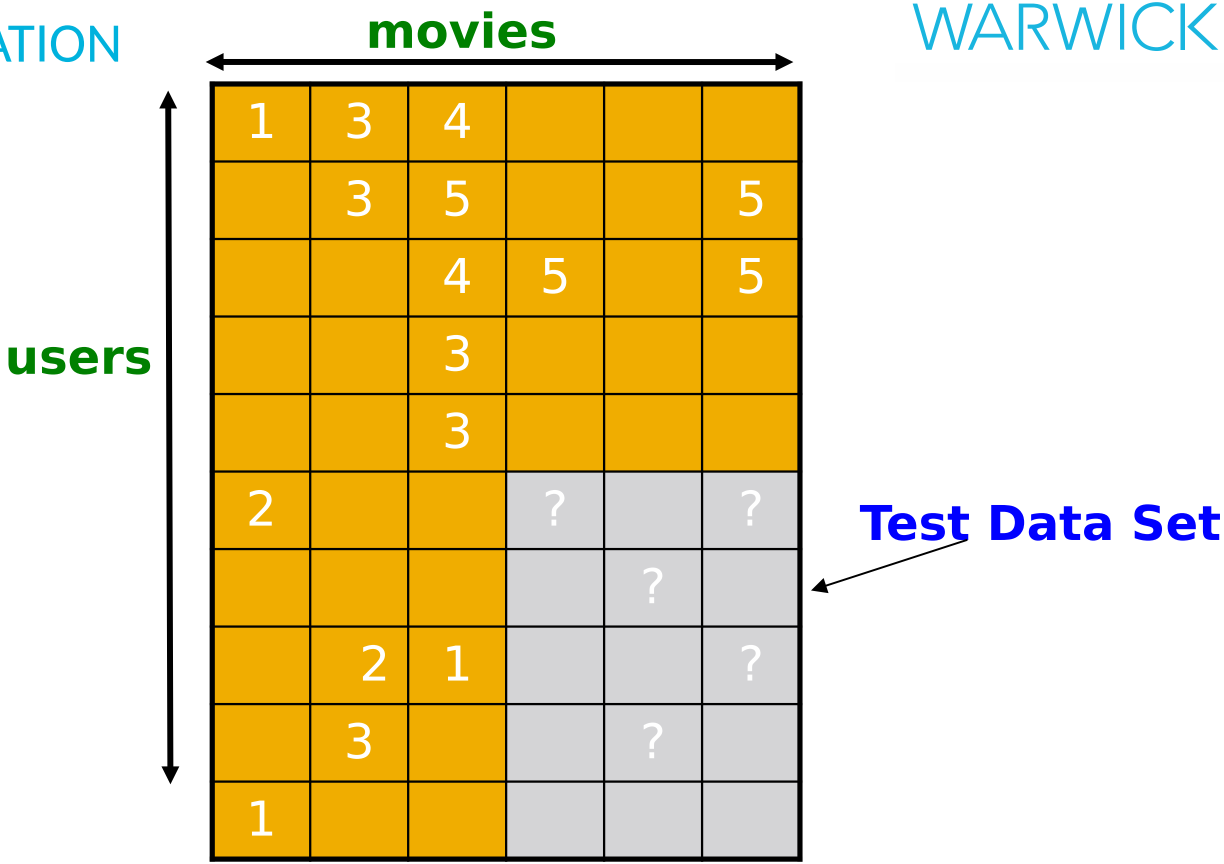
1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

DATASET FOR EVALUATION



DATASET FOR EVALUATION

Alternatively, we can hold out entire users or entire items, especially if we need to test with cold-start and/or introducing new products.



EVALUATION APPROACHES

WARWICK

- Depending on the type of recommender (ranked, rated or not) we may evaluate as:
  - **Unranked, unrated:** A classification problem (P, R, F1, accuracy).
  - **Ranked:** An information retrieval problem (MAP, NDCG).
  - **Rated:** A rating comparison problem (RMSE, MAE).

WARWICK

EVALUATING PREDICTIONS (I)

- Compare predictions with known ratings:
  - **Root-mean-square error (RMSE)**
    comparing predicted ratings vs actual ratings

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{t=1}^{n} (\hat{y}_t - y_t)^2}{n}}.$$

  - **Mean Absolute Error (MAE)**

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}.$$

i.e. how close are we from the actual ratings?

for something actually rated as 5, it's not the same to predict a 4, or to predict a 1.

EVALUATING PREDICTIONS (II)

- Compare predictions with known rankings:
  - **Evaluate as for ranking:**
    - Precision-at-K
    - MAP
    - NDCG – recall, as opposed to the 2 above, can consider ratings, not only 0/1 judgements

EVALUATING PREDICTIONS (III)

- Alternatively, **0/1 model**, no ranking/ratings considered:
  - **Coverage:**
    Number of items/users that the user recommends, i.e. we count for instance how many of the items in our database are being recommended to somebody.
  - **Precision:**
    Accuracy of predictions.

CHOOSING AN EVALUATION METRIC

WARWICK

- As usual, it **depends on the task** you have at hand!

- **General advice:**
  - If we're recommending just 1-3 items, then precision might suffice.
  - If we're recommending more items, then ranking metrics are more useful.
    - If those items have ratings, then NDCG, RMSE or MAE.

MORE EVALUATION

- There are some **aspects that are specific to recommender systems**.
  - We need something **beyond standard classification and IR evaluation metrics.**

- Important aspects in recommender systems:
  - Novelty.
  - Serendipity.
  - Diversity.

MORE EVALUATION: NOVELTY

- **Question:** are we recommending items that the user was not aware of, or they have not seen before?
- **How to measure:** online experimentation, explicitly asking users if they were aware of a recommended item.

MORE EVALUATION: SERENDIPITY

WARWICK

- **Serendipity** is a stronger condition than novelty.
- **Question:** are we making recommendations that are not obvious?

- Two ways of evaluating:
  - Online evaluation.
  - Offline evaluation.

MORE EVALUATION: SERENDIPITY

WARWICK

- **Online evaluation**, explicitly asking the user:
  - Was the recommendation useful?
  - Was the recommendation non-obvious?

- Where the user responds (1) useful and (2) non-obvious, then we can consider that to be a serendipitous finding.

- We can then compute the ratio of serendipitous vs non-serendipitous findings.

MORE EVALUATION: SERENDIPITY

- **Offline evaluation**, an approximation to evaluate our system A:
  - Build a naive content-based recommender system B, prone to recommend obvious items (i.e. very similar in content).
  - Then, evaluating our system A, compute the number of correct recommendations that are not recommended by B (i.e. non-obvious). This gives us an estimate of the ratio of serendipitous recommendations.

MORE EVALUATION: DIVERSITY

WARWICK

- **Question:** are we recommending different types of items? e.g. if we recommend 10 songs, are they from different bands?

- **How to measure:** content-based similarity metric between pairs of items. The average of pairwise similarities between recommended items can be reported as the diversity.
  - The more dissimilar the items we recommend, the higher diversity.

ASSOCIATED READING

- Aggarwal, Charu C. Recommender systems. Springer International Publishing, 2016. **Chapter 4 (content-based) and Chapter 7 (evaluation)**. https://link.springer.com/content/pdf/10.1007/978-3-319-29659-3.pdf