Content-Based Recommendation Systems

Objectives

- Define recommendation system
- Differentiate between content-based and collaborative filtering
- Introduce the main components of content-based recommenders
    - Pre-processing and feature extraction
    - Content-based learning of user profiles
    - Filtering and recommendation
- Build your own content-based recommender (assignment)

Recommendation systems

A recommender is an information filtering and retrieval system system whose purpose is to provide useful recommendations to the user (usually in a user-item paradigm).

There are generally two types of recommenders:

Content-based
Item attributes and user-item preferences are used to recommend similar items.

Collaborative filtering
User-item interactions (such as ratings) are used to identify similarities between users and ratings to recommend items.

Hybrid
Combinations of these approaches are common.

Examples of content-based recommenders

- Mountain ski run recommender
  - You tell the recommender a trail you like, then it recommends similar trails to you.
  - A trail is a vector of features, and trail vectors similar to the trail vector the user provides are recommended.
- Pandora
  - You tell Pandora a station/artist/album you like, and then it provides similar content.

In both cases you must decide what the features are and how they should be weighted. **There is almost always significant tuning associated with making a useful content-based recommender.**

Examples of collaborative filtering recommenders

- Last.fm
  - Observes what songs you listen to on a regular basis, and also knows what other users listen to on a regular basis.
  - Are there other users that listen to the same things that I do, but also listen to things that I don't? If we agree on things we've both listened to/rated, them maybe we agree on things that haven't been rated in common.
- Netflix
  - Through ratings (explicit) and watching/not watching items (implicit), a large database of user-item interactions is created. Use these user-item interactions to figure out what to recommend, whether it's model based (e.g. matrix factorization) or neighbor-based (knn). Surely a hybrid system.

Building the Content-Based Recommender

- Build an **Item Profile** for each item (movie, article, user, ski run, etc):
  - Feature extraction
  - Feature representation and cleaning
  - Feature representation and cleaning
- Build a **User Profile** for the user:
  - Feature selection and weighting
  - Getting user preference
- Generate **Recommendations:**
  - Decide on Distance Metric
  - Return *n*-nearest items

Item Profile: Feature Extraction

From all the item attributes, extract features.

Examples:

- Bag-of-words
- Title, author
- Color
- Price

	clancy	rowling	hillenbrand	fantasy	drama	fiction	nonfiction	military	navy
Rainbow Six	1	0	0	0	1	1	0	1	0
HP Philosophers Stone	0	1	0	1	0	1	0	0	0
The Hunt for Red October	1	0	0	0	1	1	0	1	1
HP Deathly Hollows	0	1	0	1	1	1	0	0	0
Unbroken: A World War II Story of Survival	0	0	1	0	1	0	1	1	1

Item Profile: Feature Representation and Cleaning

Express features in a representative vector-space

Examples:

- remove stop words
- tf or tf-idf
- phrase extraction (n-grams?)
- standardize (or not) some numerical columns

User Profile: Feature Selection and Weighting

Ensure that only the “most informative” features are maintained in the vector space representation. Decide how much to weight the features (typically a iterative, heuristic process.)

Examples:

- Does noise in the data lead to overfit recommendations?
- tf or tf-idf?
- Expand/contract stop word set?
- Re-scale numerical features
- For like/dislike add feature values for likes and subtract for dislikes
- Add more weight to newer ratings (time decay for older interactions)
- Add more weight to more relevant features (news *topic*, movie *genre*)

User Profile: Getting User Preference

Need to know what items the user likes.

Examples:

- Ratings
- Implicit feedback
- Text description (sentiment analysis)
- Cases (or items) they are interested in

Use this information to weight and create a vector representation of a user:

	clancy	rowling	hillenbrand	fantasy	drama	fiction	nonfiction	military	navy
Dan	0.3	1.0	0.0	0.0	0.0	0.6	0.0	0.7	0.0

Generate Recommendation

Knowing what item(s) the user likes, find items with similar attributes.

- Decide on a similarity metric (cosine, euclidean, jaccard)
- Similarity metric for data containing both numerical and categorical data: recommend **k prototypes** (paper [see eqn. 2.3], implementation in python)
  - See `stocks.py` in provided lecture folder
- Find the n-items that are similar
- Recommend them!

Pros/ Cons of Content-Based Recommendors

- Pros:
  - Easy to interpret reason for recommendations
  - Easy to modify recommendations by modifying the layout of item features
  - Can avoid the Cold Start problem for items -(more so than collaborative recommendors)
- Cons:
  - Recommendations will not be diverse or unexpected
  - Limited understanding of context
  - "If I like Sandra Bullock in action films and Meg Ryan in comedies, but if I hate Meg Ryan in action films and Sandra Bullock in comedies, there's no way for that to be captured in the feature vector" (Joeseph Konstan)

Objectives

- Define recommendation system
- Differentiate between content-based and collaborative filtering
- Introduce the main components of content-based recommenders
    - Pre-processing and feature extraction
    - Content-based learning of user profiles
    - Filtering and recommendation
- Build your own content-based recommender (assignment)