CS6120: Intelligent Media Systems

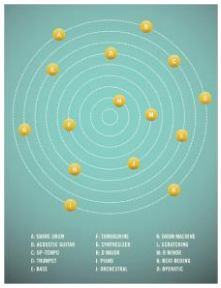
Dr. Derek Bridge
School of Computer Science & Information Technology
UCC

Content-Based Recommenders

- Content-based recommenders require *descriptions*
- Item descriptions
  - assume, for now, each description is just a set of keywords or *terms*
  - e.g. for movies: genre (action, comedy,…) but maybe actor and director names, maybe words that describe the plot
  - other terminology: tags, metadata
- User profile
  - also a set of keywords or terms that describe a user's tastes

Item Descriptions

- Tagging an item
  - means describing it using a set of terms
- Experts using a controlled vocabulary
  - a fixed, agreed set of terms
  - e.g. Pandora's Music Genome Project
  - e.g. Netflix employs 40 people to hand-tag TV shows and movies



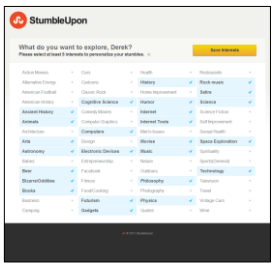
New York Time's article: The Song Decoders

Item Descriptions

- End-users using an open vocabulary
  - e.g. flickr, last.fm,…
- Why will they do this?
  - to organize their collection
  - to make it amenable for search
  - self-expression,…
- But can be very noisy

User Profiles

- Explicit
  - ask user to supply keywords when s/he registers
- Implicit
  - add keywords to user profile from item descriptions of items in which s/he shows an interest



Similarity

- The similarity of set of terms A and set of terms B
  - e.g. A = a user profile and B = an item description
  - e.g. A = an item description and B = another item description
- Can be based on the size of their intersection:

$$|A \cap B|$$

- But this gives bigger sets a greater chance of higher similarity
- Better is *Jaccard similarity*, which normalizes by the size of the union:

$$\frac{|A \cap B|}{|A \cup B|}$$

Example

Title	Director	...	Terms
Skyfall	Mendes	...	spy, action, train, revenge, violence
Pulp Fiction	Tarantino	...	boxer, violence, restaurant, dancing
Tinker, Tailor...	Alfredson	...	intelligence, spy, identity, mole
Casino Royale	Campbell	...	terrorism, spy, violence, banker
Batman Begins	Nolan	...	revenge, butler, identity

- How similar are
  - *Skyfall* and *Casino Royale*?
  - *Tinker, Tailor* and *Batman Begins*?

Term Vectors

- We can instead represent the sets as binary vectors
- Each position corresponds to one of the terms
- E.g.

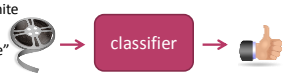
	action	banker	boxer	butler	dancing	identity	intelligence	mole	restaurant	revenge	spy	terrorism	train	violence
Skyfall	1	0	0	0	0	0	0	0	0	1	1	0	1	1
Pulp Fiction	0	0	1	0	1	0	0	0	1	0	0	0	0	1

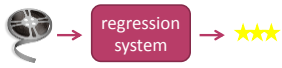
- Compute Jaccard similarity efficiently using vector operations

$$\frac{A \cdot B}{\|A\|^2 + \|B\|^2 - A \cdot B}$$

Two Kinds of Prediction

- In AI/ Machine Learning/ Data Mining
  - Classification
    - predict one of a small, finite set of labels
    - in our case, "like", "dislike"
  - Regression
    - predict a number
    - in our case, e.g., 1-5





It's Time to Build a Recommender!

- So, we have
  - item descriptions
  - user profiles
  - a way of measuring similarity
  - a set of candidate items
- For each candidate item, the recommender must make a prediction

A classifier

for each candidate item, i

compute $sim(i, u)$, similarity between item description and user profile

if $sim(i, u) \geq \theta$

predict "like" for item i

else

predict "dislike" for item i

recommend the candidates for which the prediction is "like" in descending order of similarity

Example

Title	Director	...	Terms	*sim*	like?
Skyfall	Mendes	...	spy, action, train, revenge, violence	3/6	like
Pulp Fiction	Tarantino	...	boxer, violence, restaurant, dancing	1/7	dislike
Tinker, Tailor...	Alfredson	...	intelligence, spy, identity, mole	2/6	like
Casino Royale	Campbell	...	terrorism, spy, violence, banker	2/6	like
Batman Begins	Nolan	...	revenge, butler, identity	1/6	dislike

User	Terms
Ann	spy, action, violence, identity

- If $\theta = 1/3$,
  - recommend "Skyfall", "Tinker, Tailor,..." and "Casino Royale"
    - in that order

Lifting the Burden

Problems

- Item descriptions
  - expert tagging is expensive
  - end-user tagging is 'uneven'
- User profiles
  - explicit construction on registration is inconvenient and gives profiles that are static

Solutions

- Item descriptions
  - extract terms from, e.g.:
    - book or movie plot synopsis
    - review of a restaurant,…
    - the text itself of web pages, news stories,…
- User profiles
  - use terms from items s/he likes

Term Frequency Vectors

- Suppose we're extracting the terms from a 'document'
- Instead of 0/1, we can use numeric *weights* that reflect the importance of a keyword
- Simplest is *term frequency*, tf_t
  - how often term t appears in the document
- E.g.

	action	banker	boxer	butler	dancing	identity	intelligence	mole	restaurant	revenge	spy	terrorism	train	violence
Skyfall	1	0	0	0	0	0	0	0	0	3	5	0	2	1
Pulp Fiction	0	0	4	0	1	0	0	0	2	0	0	0	0	1

Problem with Term Frequency

- Problem
  - some words will be frequent but not important
  - e.g. in a plot synopsis for a film "man", "woman",…
  - one sign of this is that they occur in many of the documents

Inverse Document Frequency

- Let df_t be the *document frequency* of term t
  - how many documents t occurs in
- The importance of t is inversely related to df_t
- The inverse document frequency of t is given by

$$idf_t = \frac{N}{1+ df_t}$$

where N is the total number of documents

TF-IDF

- So the weights we use in our term vectors are based on combining
  - term frequency (i.e. within the document)
  - inverse document frequency (i.e. across documents)

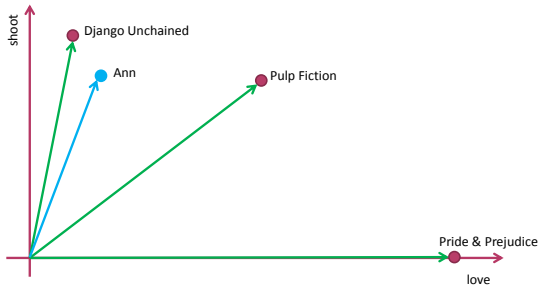
$$tfidf_t = tf_t \times idf_t$$

TFIDF: niceties

- There are numerous variations
  - sublinear tf scaling
  - maximum tf normalization
  - log scaling of idf
  - cosine normalization or pivot normalization
- And we use Cosine Similarity

$$\frac{A \cdot B}{\|A\| \times \|B\|}$$

Vector Space Model



Problem

- The vectors can be very long
- Solutions:
  - Preprocessing
    - ignore stop words
      - e.g. "a", "an", "the", "in",…
      - e.g. the list used by SQL
    - stem
      - e.g. "laughs", "laughing", "laughed" → "laugh"
      - e.g. the Porter Stemming Algorithm
  - Postprocessing
    - keep only the most discriminating terms

More Problems

- This representation loses relationships between words in the text, e.g. their order, contiguity
  - E.g. the following are regarded as similar
    - "The secret agent jumps from the moving train…"
    - "The bride's secret conversation with her travel agent…"
- Ambiguity may lead to false matches
  - e.g. different documents might be using different meanings of "bank", "train", "fire"
- Synonymy may lead to missed matches
  - e.g one document might use "spying"; another might use "espionage"

More Problems

- Lack of domain knowledge may lead to missed matches
  - e.g. one document might discuss "impressionism" another discusses "Claude Monet"
- The semantics (meaning) is not being captured
  - E.g. consider this review of a steak house
    - *"There is nothing on the menu to suit a vegetarian…"*

    with these two
    - *"This is a meat-lover's paradise…"*
    - *"The vegetarian options are superb…"*
  - Which is the more similar?

The Same Recommender

for each candidate item, i

compute $sim(i, u)$, similarity between item description and user profile

if $sim(i, u) \geq \theta$

predict "like" for item i

else

predict "dislike" for item i

recommend the candidates for which the prediction is "like" in descending order of similarity

It's dynamic!

- One nice characteristic
  - user profile is not static:
    - each time s/he shows interest in an item, that item's terms are used to update the user profile
  - e.g. after she watches *Skyfall* (or gives it a high rating), we update her profile using *Skyfall*'s term vector

	action	banker	boxer	butler	dancing	identity	intelligence	mole	restaurant	revenge	spy	terrorism	train	violence
Skyfall	1	0	0	0	0	0	0	0	0	3	5	0	2	1

Problem

- Ben likes nature documentaries 
- He likes war movies 
- But he doesn't like war documentaries 

- But his profile will have high *tfidf* scores for nature, documentary and war
  - from nature documentaries and war movies that he liked
- So a war documentary will be similar and may be recommended

Solution

- Instead of a single vector, which mashes together terms from all items s/he likes…
- …let the profile be a set of vectors, one for each item, along with her/his rating of the item
- E.g. if Clare likes *Skyfall* (4 stars) and *Pulp Fiction* (5 stars) but not *Batman Begins* (2 stars):

	action	banker	boxer	butler	dancing	identity	intelligence	mole	restaurant	revenge	spy	terrorism	train	violence	RATING
Skyfall	1	0	0	0	0	0	0	0	0	3	5	0	2	1	4
Pulp Fiction	0	0	4	0	1	0	0	0	2	0	0	0	0	1	5
Batman Begins	0	0	0	5	0	3	0	0	0	4	0	0	0	0	2

Regression using Nearest Neighbour

for each candidate item, i

  for each item in the user's profile, j

    compute $sim(i, j)$

  let j^* be the nearest neighbour, i.e. the item in the profile that is most similar to i

  let the predicted rating for item i be j^*'s rating

recommend the candidates in descending order of predicted rating

Example

- Suppose this is Clare's profile:

	action	banker	boxer	butler	dancing	identity	intelligence	mole	restaurant	revenge	spy	terrorism	train	violence	RATING	sim
Skyfall	1	0	0	0	0	0	0	0	0	3	5	0	2	1	4	0.75
Pulp Fiction	0	0	4	0	1	0	0	0	2	0	0	0	0	1	5	0.06
Casino Royale	1	2	0	0	0	0	1	0	1	1	4	2	0	2	5	0.89
Batman Begins	0	0	0	5	0	3	0	0	0	4	0	0	0	0	2	0.16

- Suppose the first candidate item is:

Goldeneye	3	1	0	0	0	0	2	0	0	2	4	3	1	2

- *Casino Royale* is the nearest neighbour
  - Clare's rating for *Casino Royale* is 5
  - So the predicted rating for *Goldeneye* is also 5

Example

- Suppose this is Clare's profile:

	action	banker	boxer	butler	dancing	identity	intelligence	mole	restaurant	revenge	spy	terrorism	train	violence	RATING	sim
Skyfall	1	0	0	0	0	0	0	0	0	3	5	0	2	1	4	0.30
Pulp Fiction	0	0	4	0	1	0	0	0	2	0	0	0	0	1	5	0.15
Casino Royale	1	2	0	0	0	0	1	0	1	1	4	2	0	2	5	0.33
Batman Begins	0	0	0	5	0	3	0	0	0	4	0	0	0	0	2	0.40

- Suppose the second candidate item is:

Spiderman	2	0	0	0	0	3	0	0	1	1	0	1	1	1

- *Batman Begins* is the nearest neighbour
  - Clare's rating for *Batman Begins* is 2
  - So the predicted rating for *Spiderman* is also 2

Example

- Of these two candidates, we recommend
  1. *Casino Royale* (predicted rating: 5 stars)
  2. *Spiderman* (predicted rating: 2 stars)
- Of course, in reality, with more candidates, we could recommend
  - just the top few, or
  - only those whose predicted rating > 3
- But, the predictions are based on just the nearest neighbour
  - can often improve accuracy by using several neighbours

Regression using k-Nearest Neighbours

for each candidate item, i
  for each item in the user's profile, j
    compute $sim(i, j)$
  let NN be the set of k nearest neighbours, i.e. the k items in the profile that are most similar to i
  let the predicted rating for item i be the average of the ratings in NN
recommend the candidates in descending order of predicted rating

Example with $k = 3$

- Suppose this is Clare's profile:

	action	banker	boxer	butler	dancing	identity	intelligence	mole	restaurant	revenge	spy	terrorism	train	violence	RATING	sim
Skyfall	1	0	0	0	0	0	0	0	0	3	5	0	2	1	4	0.75
Pulp Fiction	0	0	4	0	1	0	0	0	2	0	0	0	0	1	5	0.06
Casino Royale	1	2	0	0	0	0	1	0	1	1	4	2	0	2	5	0.89
Batman Begins	0	0	0	5	0	3	0	0	0	4	0	0	0	0	2	0.16

- Suppose the first candidate item is:

Goldeneye	3	1	0	0	0	0	2	0	0	2	4	3	1	2

- The 3-nearest neighbours are *Casino Royale, Skyfall, Batman Begins*
  - Their ratings are 5, 4, and 2
  - So the predicted rating for *Goldeneye* is $(5 + 4 + 2)/3 = 3.67$

Example with $k = 3$

- Suppose this is Clare's profile:

	action	banker	boxer	butler	dancing	identity	intelligence	mole	restaurant	revenge	spy	terrorism	train	violence	RATING	sim
Skyfall	1	0	0	0	0	0	0	0	0	3	5	0	2	1	4	0.30
Pulp Fiction	0	0	4	0	1	0	0	0	2	0	0	0	0	1	5	0.15
Casino Royale	1	2	0	0	0	0	1	0	1	1	4	2	0	2	5	0.33
Batman Begins	0	0	0	5	0	3	0	0	0	4	0	0	0	0	2	0.40

- Suppose the second candidate item is:

Spiderman	2	0	0	0	0	3	0	0	1	1	0	1	1	1

- The 3-nearest neighbours are *Batman Begins, Casino Royale, Skyfall*
  - Their ratings are 2, 5, and 4
  - So the predicted rating for *Spiderman* is $(2 + 5 + 4)/3 = 3.67$

Example

- Using more than one neighbour is a good idea
- But here, we have ended up with the same predicted rating for both candidates:
  1. *Casino Royale* (predicted rating: 3.67 stars)
  2. *Spiderman* (predicted rating: 3.67 stars)
- The fix is to use a *weighted average*
  - weighted by similarity

$$\text{predicted rating for j} = \frac{\sum_{i \in NN} sim(i,j) \times \text{rating for } i}{\sum_{i \in NN} sim(i,j)}$$

Regression using k-Nearest Neighbours

for each candidate item, i
  for each item in the user's profile, j
    compute $sim(i,j)$
  let NN be the set of k nearest neighbours, i.e. the k items in the profile that are most similar to i
  let the predicted rating for item i be the *weighted average* of the ratings in NN
recommend the candidates in descending order of predicted rating

Example

- *Goldeneye*'s 3-nearest neighbours:

	RATING	sim
Skyfall	4	0.75
Casino Royale	5	0.89
Batman Begins	2	0.16

- The predicted rating is $\frac{0.89 \times 5 + 0.75 \times 4 + 0.16 \times 2}{0.89 + 0.75 + 0.16} = 4.3$

- *Spiderman*'s 3-nearest neighbours:

	RATING	sim
Skyfall	4	0.30
Casino Royale	5	0.33
Batman Begins	2	0.40

- The predicted rating is $\frac{0.40 \times 2 + 0.33 \times 5 + 0.30 \times 4}{0.40 + 0.33 + 0.30} = 3.5$

Discussion

- Content-based recommenders need item descriptions
- And the descriptions need to be predictive of people's tastes

- Where this works:
  - e.g. new stories, web pages
- Where this is more problematic
  - books, movies, music, pieces of art,…
  - is a plot synopsis predictive?
  - are reviews predictive?

Discussion

New items

- When a new item becomes available

  - it can be recommended immediately
  - we don't have to wait for people to rate it
  - contrast with collaborative recommenders (the *cold-start problem*)

New users

- When a new user joins,
  - no recommendations can be made to him/her until s/he has built a user profile
    - either supplying keywords when s/he registers
    - or rating some items when s/he registers
    - or rating items while using the system
  - similar to collaborative recommenders

Discussion

- Content-based recommenders tend not to extend our tastes
  - they recommend items similar to ones in our profile
  - generally, no *serendipitous recommendations*