Announcement: Project Proposal **due this Thursday** (no late periods)

Upload homework on time!

**Recommender Systems:**
Content-based Systems & Collaborative Filtering

**CS547 Machine Learning for Big Data**
Tim Althoff

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING
High Dimensional Data

High dim. data
- Locality sensitive hashing
- Clustering
- Dimensionality reduction

Graph data
- Community Detection
- Spam Detection

Infinite data
- Sampling streams
- Filtering data streams
- Queries on streams

Machine learning
- Decision Trees
- Perceptron, kNN

Apps
- Recommender systems
- Association Rules
- Duplicate document detection
Example: Recommender Systems

- **Customer X**
  - Buys Metallica CD
  - Buys Megadeth CD

- **Customer Y**
  - Does search on Metallica
  - Recommender system suggests Megadeth from data collected about customer X
Recommendations

Examples:

- Amazon.com
- Pandora
- StumbleUpon
- Netflix
- Google News
- Last.fm
- YouTube

Search

Recommendations

Items

Products, web sites, blogs, news items, …
From Scarcity to Abundance

- **Shelf space is a scarce commodity for traditional retailers**
  - Also: TV networks, movie theaters, ...

- **Web enables near-zero-cost dissemination of information about products**
  - From scarcity to abundance

- **More choice necessitates better filters:**
  - Recommendation engines
  - **Association rules:** How *Into Thin Air* made *Touching the Void* a bestseller:
    [http://www.wired.com/wired/archive/12.10/tail.html](http://www.wired.com/wired/archive/12.10/tail.html)
Sidenote: The Long Tail

Source: Chris Anderson (2004)
Physical vs. Online

Read http://www.wired.com/wired/archive/12.10/tail.html to learn more!
Types of Recommendations

- **Editorial and hand curated**
  - List of favorites
  - Lists of “essential” items

- **Simple aggregates**
  - Top 10, Most Popular, Recent Uploads

- **Tailored to individual users**
  - Amazon, Netflix, ...
Formal Model

- $X = \text{set of Customers}$
- $S = \text{set of Items}$

- **Utility function** $u : X \times S \rightarrow R$
  - $R = \text{set of ratings}$
  - $R$ is a totally ordered set
  - e.g., **1-5 stars**, real number in $[0,1]$
# Utility Matrix

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>0.5</td>
<td></td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td>0.2</td>
<td>1</td>
<td></td>
<td>0.4</td>
</tr>
<tr>
<td>David</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Key Problems

- **(1) Gathering “known” ratings for matrix**
  - How to collect the data in the utility matrix

- **(2) Extrapolating unknown ratings from the known ones**
  - Mainly interested in high unknown ratings
    - We are not interested in knowing what you don’t like but what you like

- **(3) Evaluating extrapolation methods**
  - How to measure success/performance of recommendation methods
(1) Gathering Ratings

- **Explicit**
  - Ask people to rate items
  - Doesn’t work well in practice – people don’t like being bothered
  - Crowdsourcing: Pay people to label items

- **Implicit**
  - Learn ratings from user actions
    - E.g., purchase implies high rating
  - What about low ratings?
(2) Extrapolating Utilities

- **Key problem:** Utility matrix $U$ is sparse
  - Most people have not rated most items
  - **Cold Start Problem:**
    - New items have no ratings
    - New users have no history

- **Three approaches to recommender systems:**
  - 1) Content-based
  - 2) Collaborative
  - 3) Latent factor based
Content-based Recommender Systems
Content-based Recommendations

- **Main idea:** Recommend items to customer $x$ similar to previous items rated highly by $x$

**Example:**
- **Movie recommendations**
  - Recommend movies with same actor(s), director, genre, ...
- **Websites, blogs, news**
  - Recommend other sites with “similar” content
Plan of Action

- Item profiles
  - Red Circles
  - Triangles

- User profile

- Recommend

- likes
  - build

- match
Item Profiles

- For each item, create an item profile

- Profile is a set (vector) of features
  - Movies: author, title, actor, director, ...
  - Text: Set of “important” words in document

- How to pick important features?
  - Usual heuristic from text mining is TF-IDF
    (Term frequency * Inverse Doc Frequency)
    - Term ... Feature
    - Document ... Item
Sidenote: TF-IDF

\[ f_{ij} = \text{frequency of term (feature) } i \text{ in doc (item) } j \]

\[ TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}} \]

\[ n_i = \text{number of docs that mention term } i \]
\[ N = \text{total number of docs} \]

\[ IDF_i = \log \frac{N}{n_i} \]

TF-IDF score: \[ w_{ij} = TF_{ij} \times IDF_i \]

Doc profile = set of words with highest TF-IDF scores, together with their scores

Note: we normalize TF to discount for “longer” documents
User Profiles and Prediction

- **User profile possibilities:**
  - Weighted average of rated item profiles
  - **Variation:** weight by difference from average rating for item

- **Prediction heuristic: Cosine similarity of user and item profiles)**
  - Given user profile \( x \) and item profile \( i \), estimate
    \[
    u(x, i) = \cos(x, i) = \frac{x \cdot i}{||x|| \cdot ||i||}
    \]

- How do you quickly find items closest to \( x \)?
  - Job for LSH!
Pros: Content-based Approach

- +: No need for data on other users
  - No cold-start or sparsity problems
- +: Able to recommend to users with unique tastes
- +: Able to recommend new & unpopular items
  - No first-rater problem
- +: Able to provide explanations
  - Can provide explanations of recommended items by listing content-features that caused an item to be recommended
Cons: Content-based Approach

- Finding the appropriate features is hard
  - E.g., images, movies, music
- Recommendations for new users
  - How to build a user profile?
- Overspecialization
  - Never recommends items outside user’s content profile
  - People might have multiple interests
  - Unable to exploit quality judgments of other users
Collaborative Filtering

Harnessing quality judgments of other users
Collaborative Filtering

- Consider user $x$
- Find set $N$ of other users whose ratings are “similar” to $x$’s ratings
- Estimate $x$’s ratings based on ratings of users in $N$
Finding “Similar” Users

- Let \( r_x \) be the vector of user \( x \)’s ratings
- **Jaccard similarity metric**
  - **Problem:** Ignores the value of the rating
- **Cosine similarity metric**
  - \( \text{sim}(x, y) = \cos(r_x, r_y) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||} \)
  - **Problem:** Treats some missing ratings as “negative”
- **Pearson correlation coefficient**
  - \( S_{xy} = \) items rated by both users \( x \) and \( y \)
  - \[ \text{sim}(x, y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)(r_{ys} - \bar{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \bar{r}_y)^2}} \]
  - \( \bar{r}_x, \bar{r}_y \ldots \) avg. rating of \( x, y \)

\[
\begin{align*}
r_x &= [*, _, _, *, ***] \\
r_y &= [*, _, **, **, _]
\end{align*}
\]
Similarity Metric

- **Intuitively we want:** \( \text{sim}(A, B) > \text{sim}(A, C) \)
- **Jaccard similarity:** \( \frac{1}{5} < \frac{2}{4} \)
- **Cosine similarity:** \( 0.380 > 0.322 \)
  - Considers missing ratings as “negative”
  - **Solution:** subtract the (row) mean

<table>
<thead>
<tr>
<th></th>
<th>HP1</th>
<th>HP2</th>
<th>HP3</th>
<th>TW</th>
<th>SW1</th>
<th>SW2</th>
<th>SW3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D</strong></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{sim } A, B \text{ vs. } A, C: \quad 0.092 > -0.559 \]

Notice cosine sim. is correlation when data is centered at 0.
Rating Predictions

From similarity metric to recommendations:

- Let $r_{x}$ be the vector of user $x$’s ratings
- Let $N$ be the set of $k$ users most similar to $x$ who have rated item $i$
- Prediction for item $i$ of user $x$:
  
  \[ r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi} \]

  Or even better: \[ r_{xi} = \frac{\Sigma_{y \in N} s_{xy} \cdot r_{yi}}{\Sigma_{y \in N} s_{xy}} \]

  Shorthand: $s_{xy} = \text{sim}(x, y)$

- Many other tricks possible...
Item-Item Collaborative Filtering

- So far: **User-user collaborative filtering**
- **Another view:** **Item-item**
  - For item $i$, find other similar items
  - Estimate rating for item $i$ based on ratings for similar items
  - Can use same similarity metrics and prediction functions as in user-user model

\[
\begin{align*}
  r_{xi} &= \frac{\sum_{j \in N(i; x)} S_{ij} \cdot r_{xj}}{\sum_{j \in N(i; x)} S_{ij}} \\
  S_{ij} &\quad \text{similarity of items } i \text{ and } j \\
  r_{xj} &\quad \text{rating of user } x \text{ on item } j \\
  N(i; x) &\quad \text{set items which were rated by } x \text{ and similar to } i
\end{align*}
\]
**Item-Item CF (|N|=2)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **unknown rating**
- **rating between 1 to 5**
Item-Item CF ($|N|=2$)

<table>
<thead>
<tr>
<th>movies</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>?</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- estimate rating of movie 1 by user 5
**Item-Item CF (|N|=2)**

Neighbor selection: Identify movies similar to movie 1, rated by user 5

Here we use Pearson correlation as similarity:
1) Subtract mean rating $m_i$ from each movie $i$
   
   $m_1 = (1+3+5+5+4)/5 = 3.6$

   **row 1:** [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]

2) Compute dot products between rows

<table>
<thead>
<tr>
<th>movies</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>?</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Item-Item CF (|N|=2)**

```
Compute similarity weights:
s_{1,3}=0.41, s_{1,6}=0.59
```
### Item-Item CF ($|N|=2$)

**Predict by taking weighted average:**

$$r_{1.5} = \frac{(0.41 \times 2 + 0.59 \times 3)}{(0.41 + 0.59)} = 2.6$$

$$r_{ix} = \frac{\sum_{j \in N(i; x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$
Define similarity $s_{ij}$ of items $i$ and $j$

Select $k$ nearest neighbors $N(i; x)$

- Items most similar to $i$, that were rated by $x$

Estimate rating $r_{xi}$ as the weighted average:

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i; x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i; x)} s_{ij}}$$

baseline estimate for $r_{xi}$

$$b_{xi} = \mu + b_x + b_i$$

- $\mu$ = overall mean movie rating
- $b_x$ = rating deviation of user $x$  
  = (avg. rating of user $x$) $- \mu$
- $b_i$ = rating deviation of movie $i$
## Item-Item vs. User-User

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>0.5</td>
<td>0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td>0.9</td>
<td>1</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>David</td>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
</tr>
</tbody>
</table>

- In practice, it has been observed that **item-item** often works better than **user-user**
- **Why?** Items are simpler, users have multiple tastes
Pros/Cons of Collaborative Filtering

+ Works for any kind of item
  - No feature selection needed

- Cold Start:
  - Need enough users in the system to find a match

- Sparsity:
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items

- First rater:
  - Cannot recommend an item that has not been previously rated
  - New items, Esoteric items

- Popularity bias:
  - Cannot recommend items to someone with unique taste
  - Tends to recommend popular items
Hybrid Methods

- Implement two or more different recommenders and combine predictions
  - Perhaps using a linear model

- Add content-based methods to collaborative filtering
  - Item profiles for new item problem
  - Demographics to deal with new user problem
Remarks & Practical Tips

- Evaluation
- Error metrics
- Complexity / Speed
Evaluation

movies

users

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Evaluation

The diagram illustrates a test data set for a recommendation system, where the rows represent users and the columns represent movies. The shaded cells indicate movies that the system should predict ratings for, while the non-shaded cells represent known ratings.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>?</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This data set is used to evaluate the performance of the recommendation system.
Evaluating Predictions

- **Compare predictions with known ratings**
  - **Root-mean-square error (RMSE)**
    \[
    \sqrt{\frac{1}{N} \sum_{x_i} (r_{xi} - r_{xi}^*)^2}
    \]
    - \(r_{xi}^*\) is the true rating of \(x\) on \(i\)
    - \(N\) is the number of points we are making comparisons on
  - **Precision at top 10 (or k):**
    - % of those in top 10 (or k)
  - **Rank Correlation:**
    - Spearman’s correlation between system’s and user’s complete rankings

- **Another approach: 0/1 model**
  - **Coverage:**
    - Number of items/users for which the system can make predictions
  - **Precision:**
    - Accuracy of predictions
  - **Receiver operating characteristic (ROC):**
    - Tradeoff curve between false positives and false negatives
Problems with Error Metrics

- **Narrow focus on accuracy sometimes misses the point**
  - Prediction Diversity
  - Prediction Context
  - Order of predictions

- **In practice, we care only to predict high ratings:**
  - RMSE might penalize a method that does well for high ratings and badly for others
Expensive step is finding $k$ most similar customers: $O(|X|)$

Too expensive to do at runtime

- Could pre-compute
- Pre-computation takes time $O(k \cdot |X|)$
  - $X$ ... set of customers

We already know how to do this!

- Near-neighbor search in high dimensions (LSH)
- Clustering
- Dimensionality reduction
Tip: Add Data

- Leverage all the data
  - Don’t try to reduce data size in an effort to make fancy algorithms work
  - Simple methods on large data do best

- Add more data
  - e.g., add IMDB data on genres

- More data beats better algorithms
  
  [Link](http://anand.typepad.com/datawocky/2008/03/more-data-usual.html)
On Thursday:
The Netflix prize and the Latent Factor Models
On Thursday: The Netflix Prize

- **Training data**
  - 100 million ratings, 480,000 users, 17,770 movies
  - 6 years of data: 2000-2005

- **Test data**
  - Last few ratings of each user (2.8 million)
  - Evaluation criterion: root mean squared error (RMSE)
  - Netflix Cinematch RMSE: 0.9514

- **Competition**
  - 2700+ teams
  - $1 million prize for 10% improvement on Cinematch
On Thursday: Latent Factor Models

- **Next topic:** Recommendations via Latent Factor models

The bubbles above represent products sized by sales volume. Products close to each other are recommended to each other.
Latent Factor Models (i.e., SVD++)

- Geared towards females: The Princess Diaries
- Serious: Amadeus
- The Color Purple
- Sense and Sensibility
- Ocean's 11
- The Lion King
- Independence Day
- Lethal Weapon
- Braveheart
- Geared towards males: Dumb and Dumber
- Less serious: Gus
- Dave