Databases and Information Retrieval Integration

TIETS42

Recommender Systems

Kostas Stefanidis
kostas.stefanidis@uta.fi

Autumn 2016

http://www.uta.fi/sis/tie/dbir/index.html

http://people.uta.fi/~kostas.stefanidis/dbir16/dbir16-main.html
Amazon generates 35% of their sales through recommendations.
**Schindler's List** (1993)

**Your rating:** 8.9/10

Ratings: 8.9/10 from 626,604 users
Metascore: 93/100
Reviews: 1,119 user | 161 critic | 23 from Metacritic.com

In Poland during World War II, Oskar Schindler gradually becomes concerned for his Jewish workers after witnessing their persecution by the Nazis.

**Director:** Steven Spielberg

**Writers:** Thomas Keneally (book), Steven Zaillian (screenplay)

**Stars:** Liam Neeson, Ralph Fiennes, Ben Kingsley

See full cast and crew »

**Quick Links**

- Full Cast and Crew
- Plot Summary
- Trivia
- Quotes
- Awards
- Message Board
- Release Dates
- Company Credits

**Explore More**

Facebook: 29,116 people like this. Be the first of your friends.

**Related News**

Critical Intent: Roger Ebert's Life on Film

**People who liked this also liked...**

**Gandhi** (1982)

**PG** Biography | Drama | History

Biography of Mohandas K. Gandhi, the lawyer who became the famed leader of the Indian revolts against the British rule through his philosophy of nonviolent protest.

**Add to Watchlist**

**Director:** Richard Attenborough

**Stars:** Ben Kingsley, John Gielgud, C.
Groups You May Like

- **Computer Vision and Pattern Recognition**
  - **Join** - Professional Group

- **AMMA (Amsterdam Master’s in Medical Anthropology) Alumnae**
  - **Join** - Alumni Group

- **Agile Research**
  - **Join** - Professional Group

Jobs You May Be Interested In

- **Senior Web Developer**
  - Linakis Digital - Greece

- **Information Security Consultant**
  - Trustwave - Athens, Greece

- **Enterprise Propositions & Pricing**
  - Vodafone - Athens

Companies You May Want To Follow

- [Vodafone](#)
- [CIC energieKAMPUS](#)
- [Katholieke Universiteit Leuven](#)
- [Interuniversitair Centrum voor de Wetenschappen](#)
- [Center for European Policy Studies](#)
- [METLA](#)
- [Transport & Mobility Leuven](#)
- [CWI](#)
- [TIFM](#)

Feedback I See more »


recommendations

Kostas, we found more properties like Aquila Atlantis Hotel

**Galaxy Iraklio Hotel *******
Located in Heraklion's elegant district, this 5-star hotel offers 2 gourmet restaurants, a free wellness centre and a large freshwater pool. Luxurious rooms feature balconies with pool and city views...

Most recent booking for this hotel was today at 00:40

Score from 450 reviews
**Fabulous - 8.9/10**

Total price from: € 119

---

**Capsis Astoria Hotel *******
This well-known central hotel in Heraklion is located next to the Archaeological Museum.

Most recent booking for this hotel was today at 05:57

Score from 252 reviews
**Very good - 8.2/10**

Total price from: € 75.60

---

**Atrion Hotel *****
Just a short walk from Heraklion centre and the sandy beach, Atrion offers elegant accommodation with free Internet access.

Most recent booking for this hotel was today at 12:51

Score from 458 reviews
**Very good - 8.5/10**

Total price from: € 73

---

**Lato Boutique Hotel *****
Situated opposite the old city harbour, Lato Boutique Hotel features a rooftop restaurant-bar overlooking Heraklion's Venetian Fortress.

Most recent booking for this hotel was today at 11:38

Score from 799 reviews
**Fabulous - 8.7/10**

Total price from: € 82
**Picasa**

- video streaming, online DVD, Blu-ray Disc rental

**Pandora**

- internet radio

**Netflix**

- image organizer, image viewer
Recommender Systems

Recommender systems aim at suggesting to users items of potential interest to them.

Two main steps:
- **Estimate** a rating for each item and user
- **Recommend** to the user the item(s) with the highest rating(s)

**Why recommendations?**
Why recommendations?

- **Customer/user**
  - Find interesting products/things to consume
  - Narrow down the set of choices
  - Suggest additional things
  - Help exploring the space of options
  - Discover new things
  - ...

- **Seller/provider/generator**
  - Personalized service for the user
  - Increase trust
  - Improve customer loyalty
  - Increase sales
  - Opportunities for promotion, persuasion
  - Obtain knowledge about customers
  - ...

The General Picture

Recommender systems for estimating relevance

<table>
<thead>
<tr>
<th>product</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_4$</td>
<td>0.8</td>
</tr>
<tr>
<td>$X_3$</td>
<td>0.6</td>
</tr>
<tr>
<td>$X_1$</td>
<td>0.2</td>
</tr>
<tr>
<td>$X_5$</td>
<td>0.1</td>
</tr>
<tr>
<td>$X_2$</td>
<td>0</td>
</tr>
</tbody>
</table>
The General Picture

Collaborative filtering: “ask my friends about the items they like”

<table>
<thead>
<tr>
<th>product</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_4</td>
<td>0.8</td>
</tr>
<tr>
<td>X_3</td>
<td>0.6</td>
</tr>
<tr>
<td>X_1</td>
<td>0.2</td>
</tr>
<tr>
<td>X_5</td>
<td>0.1</td>
</tr>
<tr>
<td>X_2</td>
<td>0</td>
</tr>
</tbody>
</table>
The General Picture

<table>
<thead>
<tr>
<th>product</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_4 )</td>
<td>0.8</td>
</tr>
<tr>
<td>( X_3 )</td>
<td>0.6</td>
</tr>
<tr>
<td>( X_1 )</td>
<td>0.2</td>
</tr>
<tr>
<td>( X_5 )</td>
<td>0.1</td>
</tr>
<tr>
<td>( X_2 )</td>
<td>0</td>
</tr>
</tbody>
</table>

Content-based: “show me items similar to those I previously preferred”

Recommendations Generator

items data

<table>
<thead>
<tr>
<th>description</th>
<th>price</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Recommendation Generator
- Items data

- Content-based recommendation: "show me items similar to those I previously preferred"
The General Picture

Personalization

Recommendations Generator

User profile

<table>
<thead>
<tr>
<th>product</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_4$</td>
<td>0.8</td>
</tr>
<tr>
<td>$X_3$</td>
<td>0.6</td>
</tr>
<tr>
<td>$X_1$</td>
<td>0.2</td>
</tr>
<tr>
<td>$X_5$</td>
<td>0.1</td>
</tr>
<tr>
<td>$X_2$</td>
<td>0.0</td>
</tr>
</tbody>
</table>
The General Picture

Contextualization

Recommendations Generator

<table>
<thead>
<tr>
<th>product</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_4$</td>
<td>0.8</td>
</tr>
<tr>
<td>$X_3$</td>
<td>0.6</td>
</tr>
<tr>
<td>$X_1$</td>
<td>0.2</td>
</tr>
<tr>
<td>$X_5$</td>
<td>0.1</td>
</tr>
<tr>
<td>$X_2$</td>
<td>0</td>
</tr>
</tbody>
</table>
The General Picture

Combine different mechanisms

<table>
<thead>
<tr>
<th>product</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_4$</td>
<td>0.8</td>
</tr>
<tr>
<td>$X_3$</td>
<td>0.6</td>
</tr>
<tr>
<td>$X_1$</td>
<td>0.2</td>
</tr>
<tr>
<td>$X_5$</td>
<td>0.1</td>
</tr>
<tr>
<td>$X_2$</td>
<td>0.0</td>
</tr>
</tbody>
</table>

| description | price | ...
|-------------|-------|---
| ...         | ...   | ...
| ...         | ...   | ...

friends data

user profile

user context

items data

Recommendations Generator
Two main techniques:
  o  Collaborative filtering
  o  Content-based recommendations
**Collaborative Filtering**

*Word of mouth! Use the wisdom of the crowd!*

Produce interesting suggestions for a user (filtering) by using the taste of other users (collaboration)

To make suggestions/predict missing ratings, use:
- Similar users - *user-based collaborative filtering*
- Similar items - *item-based collaborative filtering*

**Assumption:**
- Users who had similar tastes in the past, will have similar tastes in the future
User-based Collaborative Filtering

Make suggestions based on preferences of similar users
- Given a user, identify his/her k most similar users
  - Cosine similarity, Jaccard similarity
- Produce recommendations based on the items that are liked by the those k users
  - avg ratings, weighted schemes

$$r(u,i) = \frac{\sum_{u \in F_u} \text{sim}(u,u') \cdot r(u',i)}{\sum_{u \in F_u} \text{sim}(u,u')}$$

<table>
<thead>
<tr>
<th></th>
<th>1 (Titanic)</th>
<th>2 (Braveheart)</th>
<th>3 (Matrix)</th>
<th>4 (Inception)</th>
<th>5 (Hobbit)</th>
<th>6 (300)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Susan</td>
<td>5</td>
<td>?</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>Bill</td>
<td>3</td>
<td>3</td>
<td>?</td>
<td>1</td>
<td>?</td>
<td>1</td>
</tr>
<tr>
<td>Jenny</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>?</td>
<td>4</td>
</tr>
<tr>
<td>Tim</td>
<td>?</td>
<td>?</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Thomas</td>
<td>?</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>?</td>
<td>4</td>
</tr>
</tbody>
</table>

Expensive online computations
Exploit relationships between items

- Compute similarities between items
  - Cosine similarity, Jaccard similarity
- Keep for each item only the k most similar items along with their similarity scores
- Use similarities to calculate ratings for items with no scores

Other techniques cluster users and recommend items the users in the cluster closest to the active user like

*Back to this, in the context of group recommendations*
Content-based Recommendations

- Analyze data information about items (docs, music, etc.)
- Extract features for items (actors, genre, etc.)
- Recommend items with features similar to items a user likes
Cold Start Problem: An *all-time classic problem*

- What happens with new users where we have no ratings yet?
  - Recommend popular items
  - Have some start-up questions (e.g., “provide 10 restaurants you love”)

- What happens with new items?
  - Content-based filtering techniques
  - Pay a set of users/customers to rate them (crowdsourcing)
Topics:
  o  **Group Recommendations**
  o  Time-aware Recommendations
Group Recommendations
Group Recommendations

Recently, group recommendations have received considerable attention

- *Group recommenders make recommendations to groups of users!*

**Restaurants** – for a work group lunch!

**Movies** – for a family!

**Places to visit** – using a travel agency!
Recently, group recommendations have received considerable attention

• *Group recommenders make recommendations to groups of users!*

So, an item must be acceptable by all the members of the group

• Use consensus functions to characterize how much the item satisfies the group as a whole

E.g., estimate group consensus based on disagreement and relevance

• **Disagreement** captures the differences in the item ratings between group members

• **Relevance** corresponds, for example, to the average of the item ratings by all group members, or, to the highest or lowest item rating by a group member
Group Recommendations

- Group recommendations following the collaborative filtering approach
  - Aggregate single user recommendations into group recommendations
  - Use different aggregation designs
- Leverage the power of a top-k algorithm for efficient aggregation

- How to locate the similar users to a given one?
  - Naïve approach: search the whole database of users
  - Clustering approach: model the user-item interactions in terms of clustering and use the extracted clusters for predictions
Group Recommendations: Outline

- Personal recommendations
- Group recommendations
- Group recommendations computation
- Experiments
- Conclusions
$I = \{i_1, i_2, ..., i_d\}$: set of items e.g., movies, books, restaurants
$U = \{u_1, u_2, ..., u_n\}$: set of users
$\text{preference}(u, i) \in [0,1]$: the preference/rating of user $u \in U$ for item $i \in I$
$Z_i$: set of users that have expressed a preference for $i$

But, typically users rate only a few items (and $|I|$ is too high!)

For an unrated item $i$, estimate its relevance for a user $u$
  - $\text{relevance}(u, i)$
How to estimate \( relevance(u,i) \)?

- **Collaborative filtering idea**: use preferences of similar users to \( u \) to produce relevance scores for unrated items of \( u \)

- Similarity is estimated in terms of some similarity/distance function
- \( F_u \): the set of similar users to \( u \), or \textit{friends}

**Definition 1 (Friends).** Let \( \mathcal{U} \) be a set of users. The friends \( \mathcal{F}_u, \mathcal{F}_u \subseteq \mathcal{U} \), of a user \( u \in \mathcal{U} \) is a set of users, such that, \( \forall u' \in \mathcal{F}_u, \text{sim}_U(u,u') \geq \delta \) and \( \forall u'' \in \mathcal{U} \setminus \mathcal{F}_u, \text{sim}_U(u,u'') < \delta \), where \( \delta \) is a threshold similarity value.
Use preferences of similar users to $u$ to estimate preferences for items unrated by $u$.

Other than similarity?
Which is the appropriate set of users for computing the recommendations of a user?
Use preferences of similar users to \( u \) to estimate preferences for items unrated by \( u \)

Other than similarity?
Which is the appropriate set of users for computing the recommendations of a user?

Three different aspects of peers:

- **similar users**
  - When using a trip advisor, the choice of users with similar tastes seems appropriate

- **close friends**: similar tastes for most things, because of the closeness of relationship
  - When asking for a suggestion about a movie, the user’s close friends may provide good answers

- **domain experts**: the experts for the domain of a query
  - When asking for advice for a PC, domain experts may fit well to the user needs
Personal Recommendations

- Relevance computation based on friends

\[ relevance(u, i) = \frac{\sum_{u' \in (F_u \cap P_i)} \text{sim}_U(u, u') \text{preference}(u', i)}{\sum_{u' \in (F_u \cap P_i)} \text{sim}_U(u, u')} \]

- But, how confident are the relevance scores? (sparsity!)

\[ support(u, i) = \frac{|F_u \cap P_i|}{|F_u|} \]

- To estimate the worthiness of an item recommendation, combine relevance and support scores:

**Definition 2 (Personal Value).** Let \( U \) be a set of users and \( I \) be a set of items. Let \( w_1, w_2 \geq 0 : w_1 + w_2 = 1 \). The personal value of an item \( i \in I \) for a user \( u \in U \) with friends \( F_u \), such that, \( \#\text{preference}(u, i) \), is:

\[ \text{value}_{F_u}(u, i) = w_1 \times relevance(u, i) + w_2 \times support(u, i) \]
Group Recommendations: Outline

- Personal recommendations
- **Group recommendations**
- Group recommendations computation
- Experiments
- Conclusions
Group Recommendations

What is the recommendation score of an item \( i \in I \) for a group of users \( G=\{u_1, u_2, \ldots, u_k\} \subseteq \mathbb{U} \)?

- Aggregate the personal scores of the group members into overall group recommendation scores

**Definition 3 (Group Value).** Let \( \mathbb{U} \) be a set of users and \( \mathcal{I} \) be a set of items. Given a group of users \( \mathcal{G}, \mathcal{G} \subseteq \mathbb{U} \), the group value of an item \( i \in \mathcal{I} \) for \( \mathcal{G} \), such that, \( \forall u \in \mathcal{G}, \#\text{preference}(u, i) \), is:

\[
value(\mathcal{G}, i) = \text{Aggr}_{u \in \mathcal{G}}(value_{F_u}(u, i))
\]
Aggregation Designs

- **Least misery design**: Strong member preferences act as veto
  - e.g., do not recommend steakhouses if a vegetarian is in the group
  \[
  value(G, i) = \min_{u \in G} (value_{F_u}(u, i))
  \]

- **Most optimistic design**: The most satisfied member is the influential
  - e.g., recommend a movie to the group if a member is highly interested in it and the others are reasonable satisfied
  \[
  value(G, i) = \max_{u \in G} (value_{F_u}(u, i))
  \]

- **Fair design**: Democracy wins
  - e.g., recommend a holiday destination if on average the group is satisfied
  \[
  value(G, i) = (\sum_{u \in G} value_{F_u}(u, i)) / |G|
  \]
Given a group $G$, provide $k$ suggestions for items that are:

- Highly relevant to the preferences of all group members and
- Exhibit high support

**Definition 4. (Top-k Group Recommendations).** Let $U$ be a set of users and $I$ be a set of items. Given a group of users $G$, $G \subseteq U$, and an aggregation method $\text{Aggr}$, recommend to $G$ a list of items $I_G = \langle i_1, \ldots, i_k \rangle$, $I_G \subseteq I$, such that:

(i) $\forall i_j \in I_G, u \in G$, $\exists \text{preference}(u, i_j)$,
(ii) $\text{value}(G, i_j) \geq \text{value}(G, i_{j+1})$, $1 \leq j \leq k - 1$, $\forall i_j \in I_G$, and
(iii) $\text{value}(G, i_j) \geq \text{value}(G, x_y)$, $\forall i_j \in I_G$, $x_y \in I \setminus I_G$. 
Group Recommendations: Outline

- Personal recommendations
- Group recommendations
  - Group recommendations computation
- Experiments
- Conclusions
Group Recommendation Computation

(1) Locate the friends $F_u$ for all users $u \in G$

(2) Compute the personal value scores for all users $u \in G$

(3) Combine the independent scores w.r.t. $Aggr()$ and derive the top-$k$ group value scores for $G$
Group Recommendation Computation

Database

Friends Generator

Personal Recommendations Generator

Group Recommendations Generator

gRecs Engine

User Interface

Naive approach

Database

Clustering approach

$C_1$

$C_2$

$C_m$

...
(1) Friends Generator

- **Baseline approach:**
  - For a \( u \in G \), \( F_u \) consists of all his similar users in \( U \)
    
    i.e., \( F_u = \{ u' \in U : \text{sim}_U (u,u') \geq \delta \} \)
    
    ▪ No pre-computations required
    ▪ Inefficient in large systems

- **User clustering approach:**
  - Organize users into clusters of similar users
  - For a user \( u \in G \), \( F_u \) consists of the members of its corresponding cluster \( C \)
    
    i.e., \( F_u = \{ u' \in C \} \)
    
    ▪ Pre-computed groups
    ▪ Faster computations
User Clustering Approach

Use an agglomerative hierarchical clustering algorithm

- Each user is placed in his own cluster
- At each step, the two most similar clusters are merged
  - Complete link distance:
    similarity between two clusters is the min similarity between any two users in the clusters
- Stop, if the similarity of the closest pair of clusters violates the user similarity threshold $\delta$

**Property:** For each pair of users $u, u' \in C$, $\text{sim}_U(u, u') \geq \delta$

- No false positives, true negatives possible
For each user \( u \in G \), and for his unrated items \( i' \), use collaborative filtering to compute \( value(u, i') \)

Two possible implementations depending on the friends generator step (1):

- \( value_{F_u}(u, i') \) : for the baseline approach
- \( value_{C_u}(u, i') \) : for the user clustering approach

\((item, value)\) pairs are generated for each user \( u \in G \rightarrow V_u \)
(3) Top-k Group Recommendations Generator

Naïve approach:
• Aggregate $V_u$ for all $u \in G$ and compute the group value scores for all $i \in I$
• Rank the scores and report the top-$k$ valued items

A faster approach!
• TA algorithm [Fagin01]
A faster approach: TA algorithm [Fagin01]

- Use the ranked sets $V_u$
  - Two types of item access: sorted, random
- Do sorted access to each $V_u$
  - For each item seen, do random accesses to the other ranked sets to retrieve the missing personal scores
- Compute the group score of each item that has been seen
  - Rank the items based on their group scores and select the top-$k$ ones
- Stop to do sorted accesses when the group scores of the $k$ items are at least equal to a threshold
  - Threshold is the aggregation score of the scores of the last items seen in each ranked set
Access the elements sequentially

<table>
<thead>
<tr>
<th></th>
<th>$R_1$</th>
<th></th>
<th>$R_2$</th>
<th></th>
<th>$R_3$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>1</td>
<td></td>
<td>$X_2$</td>
<td>0.8</td>
<td></td>
<td>$X_4$</td>
</tr>
<tr>
<td>$X_2$</td>
<td>0.8</td>
<td></td>
<td>$X_3$</td>
<td>0.7</td>
<td></td>
<td>$X_3$</td>
</tr>
<tr>
<td>$X_3$</td>
<td>0.5</td>
<td></td>
<td>$X_1$</td>
<td>0.3</td>
<td></td>
<td>$X_1$</td>
</tr>
<tr>
<td>$X_4$</td>
<td>0.3</td>
<td></td>
<td>$X_4$</td>
<td>0.2</td>
<td></td>
<td>$X_5$</td>
</tr>
<tr>
<td>$X_5$</td>
<td>0.1</td>
<td></td>
<td>$X_5$</td>
<td>0.1</td>
<td></td>
<td>$X_2$</td>
</tr>
</tbody>
</table>
At each sequential access

- Set the threshold $t$ to be the aggregate of the scores seen in this access

<table>
<thead>
<tr>
<th></th>
<th>$R_1$</th>
<th></th>
<th>$R_2$</th>
<th></th>
<th>$R_3$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>1</td>
<td></td>
<td>$X_2$</td>
<td>0.8</td>
<td>$X_4$</td>
<td>0.8</td>
</tr>
<tr>
<td>$X_2$</td>
<td>0.8</td>
<td></td>
<td>$X_3$</td>
<td>0.7</td>
<td>$X_3$</td>
<td>0.6</td>
</tr>
<tr>
<td>$X_3$</td>
<td>0.5</td>
<td></td>
<td>$X_1$</td>
<td>0.3</td>
<td>$X_1$</td>
<td>0.2</td>
</tr>
<tr>
<td>$X_4$</td>
<td>0.3</td>
<td></td>
<td>$X_4$</td>
<td>0.2</td>
<td>$X_5$</td>
<td>0.1</td>
</tr>
<tr>
<td>$X_5$</td>
<td>0.1</td>
<td></td>
<td>$X_5$</td>
<td>0.1</td>
<td>$X_2$</td>
<td>0</td>
</tr>
</tbody>
</table>

$t = 2.6$
At each sequential access

- Do random accesses and compute the score of the objects seen

<table>
<thead>
<tr>
<th></th>
<th>R₁</th>
<th></th>
<th>R₂</th>
<th></th>
<th>R₃</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X₁</td>
<td>1</td>
<td></td>
<td>X₂</td>
<td>0.8</td>
<td>X₄</td>
<td>0.8</td>
</tr>
<tr>
<td>X₂</td>
<td>0.8</td>
<td></td>
<td>X₃</td>
<td>0.7</td>
<td>X₃</td>
<td>0.6</td>
</tr>
<tr>
<td>X₃</td>
<td>0.5</td>
<td></td>
<td>X₁</td>
<td>0.3</td>
<td>X₁</td>
<td>0.2</td>
</tr>
<tr>
<td>X₄</td>
<td>0.3</td>
<td></td>
<td>X₄</td>
<td>0.2</td>
<td>X₅</td>
<td>0.1</td>
</tr>
<tr>
<td>X₅</td>
<td>0.1</td>
<td></td>
<td>X₅</td>
<td>0.1</td>
<td>X₂</td>
<td>0</td>
</tr>
</tbody>
</table>

$t = 2.6$

X₁ | 1.5
X₂ | 1.6
X₄ | 1.3
At each sequential access

- Maintain a list of top-$k$ objects seen so far

<table>
<thead>
<tr>
<th></th>
<th>$R_1$</th>
<th></th>
<th>$R_2$</th>
<th></th>
<th>$R_3$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>1</td>
<td></td>
<td>$X_2$</td>
<td>0.8</td>
<td>$X_4$</td>
<td>0.8</td>
</tr>
<tr>
<td>$X_2$</td>
<td>0.8</td>
<td></td>
<td>$X_3$</td>
<td>0.7</td>
<td>$X_3$</td>
<td>0.6</td>
</tr>
<tr>
<td>$X_3$</td>
<td>0.5</td>
<td></td>
<td>$X_1$</td>
<td>0.3</td>
<td>$X_1$</td>
<td>0.2</td>
</tr>
<tr>
<td>$X_4$</td>
<td>0.3</td>
<td></td>
<td>$X_4$</td>
<td>0.2</td>
<td>$X_5$</td>
<td>0.1</td>
</tr>
<tr>
<td>$X_5$</td>
<td>0.1</td>
<td></td>
<td>$X_5$</td>
<td>0.1</td>
<td>$X_2$</td>
<td>0</td>
</tr>
</tbody>
</table>

$t = 2.6$

| $X_2$ | 1.6 |
| $X_1$ | 1.5 |
At each sequential access

- When the scores of the top-k are greater or equal to the threshold, stop

<table>
<thead>
<tr>
<th>R₁</th>
<th>R₂</th>
<th>R₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₁</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>X₂</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>X₃</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>X₄</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>X₅</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>X₂</td>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td>X₃</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>X₁</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>X₄</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>X₅</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>X₁</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>X₃</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>X₅</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>X₂</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

t = 2.1
At each sequential access

- When the scores of the top-k are greater or equal to the threshold, stop
Return the top-k seen so far

<table>
<thead>
<tr>
<th></th>
<th>R₁</th>
<th>R₂</th>
<th>R₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₁</td>
<td>1</td>
<td>X₂</td>
<td>X₄</td>
</tr>
<tr>
<td>X₂</td>
<td>0.8</td>
<td>X₃</td>
<td>X₃</td>
</tr>
<tr>
<td>X₃</td>
<td>0.5</td>
<td>X₁</td>
<td>X₁</td>
</tr>
<tr>
<td>X₄</td>
<td>0.3</td>
<td>X₄</td>
<td>X₅</td>
</tr>
<tr>
<td>X₅</td>
<td>0.1</td>
<td>X₅</td>
<td>X₂</td>
</tr>
</tbody>
</table>

Monotonicity property: any object not seen, has score less than the threshold

- \( f(X_5) \leq t \leq f(X_2) \)
Explanations

Explain the reason behind recommendations
**Explanations**

*Explain the reason behind recommendations*

- Except for the top-\(k\) suggested items, provide an explanation of why a specific item appears in the top-\(k\) list (*the success of recommendation!*)

- Explanation template:


  \[
  \text{ITEM } i \text{ HAS GROUP VALUE SCORE } \text{value}(G,i) \text{ BECAUSE OF USER(S) } \{u_1, \ldots, u_y\}
  \]

  - e.g., “Movie Dracula has group value score 0.9 because of user Jeffrey”

- Explanations depend on the aggregation design:
  - **Least misery**: for each suggestion, the person with the *min* personal score is reported
  - **Most optimistic design**: for each suggestion, the person with the *max* personal score is reported
  - **Fair design**: for each suggestion, the members of the group *close* to the average value are reported
Group Recommendations: Outline

- Personal recommendations
- Group recommendations
- Group recommendations computation
- Experiments
- Conclusions
Experiments

**Goal:** evaluate user clustering vs baseline approach

- MovieLens dataset (1,000 users; 1,700 items; 100,000 ratings)
- Evaluation criteria
  - Quality of recommendations
    - commonRecs: # common suggested items by both approaches
    - rankRecsDist: distance between two partial rankings based on # of pairwise disagreements between them
  - Efficiency of recommendations
    - Compare the execution time for computing recs
- To set up a query group, we randomly select the members of the group
- We run each experiment 100 times and report avg values
Time complexity for fair design with $w_1=0.5$, $w_2=0.5$

- User clustering requires almost 25% of the time required by baseline approach.
- For larger $|G|$, reduction becomes more evident.
Quality of Recommendations: Fair Design (w1 = w2 = 0.5)

commonRecs

rankRecsDist
Quality of Recommendations Overview

- As $|G|$ increases, commonRecs score decreases
  - Group recommendations rely in a more diverse set of users and personal values
- Accordingly, rankRecsDist increases as $|G|$ increases
- The fair and the least misery designs achieve better results when compared to the most optimistic design
  - Since the members of the query group are selected randomly, this is expected, since it is more difficult to find agreements for max personal values
- When $\delta$ increases, the commonRecs decreases for the fair and least misery designs and slightly increases for the most optimistic design
  - Corresponding findings also hold for rankRecsDist
Effect of weighting factors $w_1$, $w_2$

Fair design for $\delta=0.30$ with $w_1=1$, $w_2=0$

- `commonRecs` and `rankRecsDist` behave worst comparing to the equal importance case ($w_1=w_2=0.5$)
- It seems that support improves the quality of recommendations
our demo @ DASFAA 2012
What movie shall we watch tonight?

Step 1: Select your company!

- Select users from the DB
- Select users that qualify specific criteria
- Select a predefined group of users
Select a predefined group of users

Query 1
4 users, young women (21-30), students
- Select Query 1

Query 2
3 users, middle-aged women (41-50), educators
- Select Query 2

Query 3
3 users, old men (61-70), retired
- Select Query 3

Query 4
4 users, young men (21-30), engineers
- Select Query 4
**our demo @ DASFAA 2012**

**What movie shall we watch tonight?**

<table>
<thead>
<tr>
<th>Movie</th>
<th>Score</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>One Flew Over the Cuckoo's Nest (1975)</td>
<td>0.938</td>
<td>Due to users: 844 (score 0.878), 527 (score 1.0)</td>
</tr>
<tr>
<td>Full Monty, The (1997)</td>
<td>0.911</td>
<td>Due to users: 714 (score 0.921), 527 (score 0.95)</td>
</tr>
<tr>
<td>L.A. Confidential (1997)</td>
<td>0.898</td>
<td>Due to users: 714 (score 0.934), 746 (score 0.877)</td>
</tr>
<tr>
<td>Streetcar Named Desire, A (1951)</td>
<td>0.875</td>
<td>Due to users: 746 (score 1.0), 714 (score 0.76)</td>
</tr>
<tr>
<td>Persuasion (1995)</td>
<td>0.875</td>
<td>Due to users: 746 (score 0.75), 746 (score 0.75)</td>
</tr>
<tr>
<td>Boot Das (1931)</td>
<td>0.849</td>
<td>Due to users: 527 (score 0.974), 527 (score 0.75)</td>
</tr>
<tr>
<td>Psycho (1950)</td>
<td>0.844</td>
<td>Due to users: 527 (score 0.837), 527 (score 0.871)</td>
</tr>
<tr>
<td>Silence of the Lambs, The (1981)</td>
<td>0.832</td>
<td>Due to users: 714 (score 0.833), 844 (score 0.871)</td>
</tr>
<tr>
<td>To Kill a Mockingbird (1962)</td>
<td>0.813</td>
<td>Due to users: 527 (score 0.75), 714 (score 1.0)</td>
</tr>
<tr>
<td>Vertigo (1958)</td>
<td>0.813</td>
<td>Due to users: 527 (score 0.75), 746 (score 0.75)</td>
</tr>
<tr>
<td>Roman Holiday (1953)</td>
<td>0.813</td>
<td>Due to users: 746 (score 0.75), 627 (score 0.75)</td>
</tr>
<tr>
<td>Rear Window (1954)</td>
<td>0.805</td>
<td>Due to users: 844 (score 0.75), 627 (score 0.918)</td>
</tr>
</tbody>
</table>

**You can also see the individual recommendations for each user**

<table>
<thead>
<tr>
<th>User 627</th>
<th>Movie</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>627</td>
<td>Four Rooms (1995)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Crimson Tide (1995)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Eat Drink Man Woman (1994)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Three Colors: Red (1994)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Three Colors: Blue (1993)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Three Colors: White (1994)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Searching for Bobby Fischer (1993)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Big Night (1996)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Monty Python's Life of Brian (1979)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Koliva (1996)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Good Will Hunting (1997)</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User 714</th>
<th>Movie</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>714</td>
<td>Babe (1995)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Braveheart (1995)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Taxi Driver (1975)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Disclosure (1984)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Shawshank Redemption, The (1994)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Forrest Gump (1994)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Much Ado About Nothing (1993)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Aladdin (1992)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Mystery Science Theater 3000: The ..</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Lone Star (1996)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Supercop (1992)</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User 746</th>
<th>Movie</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>746</td>
<td>Usual Suspects, The (1995)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Three Colors: Blue (1993)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Gone with the Wind (1939)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Citizen Kane (1941)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>2001: A Space Odyssey (1968)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Ghost and the Darkness, The (1996)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Henry V (1980)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Cyrano de Bergerac (1990)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Room with a View, A (1986)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Full Monty, The (1997)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Rainmaker, The (1997)</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User 844</th>
<th>Movie</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>844</td>
<td>Dead Man Walking (1995)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Apollo 13 (1995)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Strange Days (1995)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Clerks (1994)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>What's Eating Gilbert Grape (1993)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Wallace &amp; Gromit, The Best of Aard...</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Godfather, The (1972)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Wizard Of Oz, The (1939)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Citizen Kane (1941)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Fish Called Wanda, A (1982)</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>On Golden Pond (1981)</td>
<td>1.0</td>
</tr>
</tbody>
</table>

**Give it another try!**
Summary for Group Recommendations

- **Focus**: Collaborative filtering for group recommendations
  - Instead of exhaustively search for similar users in the whole user base
    - *Pre-partition users into clusters of similar ones*
    - *Use the cluster members for recommendations*
  - Efficiently aggregate the single user recs into group recs
    - *Leverage the power of a top-k algorithm*

- **Results**: User clustering considerably improves the execution time, while preserves a satisfactory quality of recommendations

- **Next goal!**
  - Exploit subspace clustering
Exploring Subspace Clustering for Recommendations

Coupling recommendations computation with subspace clustering

- High dimensionality $\rightarrow$ subspace clustering
- Missing values $\rightarrow$ fault-tolerant clustering

We diversify the set of user used for recommendations

- Different friends rely upon different set of items (subspaces)

This approach outperforms traditional collaborative filtering and a full dimensional clustering approach
Time-aware Recommendations
**Time-aware Recommendations**

The general recommendation model: all ratings are equally active and potentially they can be used for producing recommendations

- Temporal aspects of user ratings are ignored

The goal: exploit the temporal information of user ratings towards improving the predictions in collaborative recommender systems

Two different types of time effects based upon:

- **Recency/freshness** of ratings
- **Temporal context** of ratings
Two types of time-aware recommendations

- **Fresh-based recommendations**: pay more attention to more recent ratings
  - Assumption: the most recent user preferences better reflect the current trends and thus, they contribute more in the computation of the recommendations

- **Temporal context-based recommendations**: offer different suggestions for different time specifications (e.g., holidays and weekdays)
  - Motivation: user preferences may change over time but have temporal repetition, i.e., recur over time

Each *preference*(u,i) is associated with the time that i was rated by u, t_{u,i}
- So, this timestamp declares the freshness or age of the rating

*The purpose here, is to provide a framework for studying various approaches that handle different temporal aspects of recommendations*
Fresh-based Recommendations

Suggest items taking mainly into account recent and novel user preferences

Different types of aging mechanisms to define the way that the historical information (in form of ratings) is incorporated in the recommendation process

- **Damped window model**: gradually decrease the importance of historical data comparing to more recent data
  - E.g., give higher priority to new releases compared to other old seasoned movies
- **Sliding window model**: remember only the preferences defined within a specific, recent time period
  - E.g., focus only on new releases
All user preferences are active, i.e., they can contribute to produce recommendations

- But, their contribution depends upon their arrival time

\( preference(u, i) \) is weighted appropriately with a temporal decay function

- Employ the exponential fading function \( 2^{-\lambda(t-t_{u,i})} \),
  - \( t \) is the current time
  - \( \lambda, \lambda > 0 \), is the decay rate that defines how fast the past history is forgotten

\[
relevance^d(u, i, Q) = \frac{\sum_{u' \in (P_{u,Q} \cap Z_i)} 2^{-\lambda(t-t_{u',i})} \times contribution(u, u') \times preference(u', i)}{\sum_{u' \in (P_{u,Q} \cap Z_i)} contribution(u, u')}
\]
Exploit only a subset of the available preferences; the most recent ones

The size of this subset, **window size**, might be defined in terms of:

- Time-points (e.g., use the preferences defined after Jan 2011) or
- Records (e.g., use the 1000 most recent preferences)

We adopt the first

The preferences within the window are the **active** preferences that participate in the recommendations computation

- Let \( t \) be the current time and \( W \) be the window size
- \( \text{preference}(u,i) \) is active only if \( t_{u,i} > t-W \)

\[
\text{relevance}^s(u,i,Q) = \frac{\sum_{u' \in (P_{u,Q} \cap \mathcal{X}_i)} \text{contribution}(u,u') \times \text{preference}(u',i)}{\sum_{u' \in (P_{u,Q} \cap \mathcal{X}_i)} \text{contribution}(u,u')} 
\]

where \( \mathcal{X}_i \) is the set of users in \( \mathcal{Z}_i \), such that, \( \forall u' \in \mathcal{X}_i, t_{u',i} > t - W \).
Temporal Context-based Recommendations

Context-based recommendations assume that preferences display some kind of temporal repetition

- *Or, in other words, users may have different preferences under different temporal contexts*

E.g.,

- A tourist guide system should provide different suggestions during summer than during winter
- A restaurant recommendation system might distinguish between *weekdays* (typically business lunches) and *weekends* (typically family lunches)
Time is modeled as a multidimensional attribute: *the dimensions of time have a hierarchical structure*

Three different levels of time:

- **time of day** with domain values {“morning”, “afternoon”, “evening”, “night”}
- **day of week** with domain values {“Mon”, “Tue”, “Wed”, “Thu”, “Fri”, “Sat”, “Sun”}
- **time of week** with domain values {“Weekday”, “Weekend”}

It is easy to derive such kind of information from the time value $t_{u,i}$ associated with each rating by using SQL or other languages
Let $\Theta$ be the current temporal context of a user $u$.

The context-based relevance of an item $i$ for $u$ under a query $Q$ expressed at $\Theta$ based on the preferences of the friends of $u$ for $i$ that are defined for the same context $\Theta$, is:

$$relevance_c(u, i, Q) = \frac{\sum_{u' \in (p_{u, Q} \cap y_i)} \text{contribution}(u, u') \times \text{preference}(u', i)}{\sum_{u' \in (p_{u, Q} \cap y_i)} \text{contribution}(u, u')}$$

$Y_i$ is the set of users in $Z_i$ with ratings expressed for a context equal to $\Theta$. 

**Temporal Context-based Recommendations**
Basic Steps of Computation

- Query submission: each query is enhanced with a contextual specification expressing some temporal information
  - E.g., look for restaurants serving Chinese cuisine during the weekend
    The user should provide the aging scheme that will be used

As above...
1. Locate the peers of the user
2. Employ their preferences for estimating the time-aware recommendations
3. Present recommendations along with explanations on the reasons behind them
Reminder

- **Damped window approach**
  - All preferences of the similar users of \( u \) are employed for computing recommendations

- **Sliding window approach**
  - Only the most recent preferences are used

- **Context-based approach**
  - Only the preferences defined for a temporal context equal to the query context are used
    - This can be seen as a preference pre-filtering step
Context-based approach

*What if the associated set of preferences for a specific query is empty?*
Context-based approach

*What if the associated set of preferences for a specific query is empty?*

- Use these preferences whose context is more general than the query context
  
  E.g., for a query with context “Sat”, use a preference defined for context “Weekend”

Do it efficiently by deploying indexes on the context of the preferences
Goal: Evaluate the effectiveness of our time-aware recommendation system

- MovieLens dataset (100,000 ratings given from September 1997 till April 1998 by 1,000 users for 1,700 items)
- Monthly split in (a), split per weekends and weekdays in (b)
Experiments

Similar users:

- **Distance instead of similarity:**
  
  - For two users, compute the Euclidean distance over the items rated by both

  $$
  \text{distU}(u, u') = \sqrt{\sum_{i \in I_u \cap I_{u'}} (\text{preference}(u, i) - \text{preference}(u', i))^2 / |I_u \cap I_{u'}|}
  $$

  $I_u$: the items for which $\exists$ preference($u, i$), $\forall i \in I_u$

  $I_{u'}$: the items for which $\exists$ preference($u', i$), $\forall i \in I_{u'}$

  $I_u \cap I_{u'}$: the items for which both users have expressed preferences
Experiments

To evaluate the quality of recommendations: directly compare the predicted ratings with the actual ones

- Mean Absolute Error (MAE): Average absolute difference between predicted ratings and actual ratings

\[ MAE = \frac{1}{N} \sum_{u,i} |preference(u, i) - value^o(u, i, Q)| \]

N is the total number of ratings in the employed dataset

The lower the MAE score, the better the predictions
Sliding Window

- Use windows of different sizes $W$
  - $W = 1$ for the most recent month, ..., $W = 8$ includes the whole dataset
- For each dataset, compute the recommendations for each user by considering the user ratings within the corresponding window $W$
- Compare the predicted values with the actual values given by the user within the same window $W$
- Report the average results

**Better quality for small windows**
- Except for the smallest one: small amount of ratings used for predictions

*For larger user distance thresholds, MAE scores increase for all window sizes*
- More dissimilar users are considered for the suggestions computation
Evaluate the effect of the decay rate $\lambda$ in the recommendations accuracy

Different values for $\lambda$
- The higher the $\lambda$ is, the less the historical data count
- $\lambda = 0$ corresponds to the time-free model

- This aging model seems to not offer any, or offer a very small, improvement in this setting, i.e., for the employed dataset
- Larger distance thresholds lead to larger MAE scores
**Temporal Context-based Recommendations**

*Demonstrate the effect of temporal context on producing recommendations*

Two different temporal contexts:

- “Weekends”, for predictions, use only ratings defined for weekends
- “Weekdays”, for predictions, consider ratings from Monday to Friday

The predicted values are compared to the actual values given by the user within the same temporal context through the MAE metric

For both, “Weekends” and “Weekdays”, the quality of the recommendations is improved compared to the time-free approach

- E.g., predictions for “Weekends” are improved on average 1.5% when using ratings for “Weekends” instead of using all ratings

The quality decreases with the distance threshold
In Overall

Time plays an important role towards improving the quality of the proposed recommendations

• The sliding window and the context-based approaches increase the accuracy
• A mere decay model seems to be not adequate

Next goal: Design a more elaborate aging scheme that considers not only the age of the ratings but also parameters, such as recency and popularity of recommended items and context under which ratings were given

The time effect will be more evident for datasets that span a larger period of time. Experimentation with other kinds of peers dataset will be interesting
Context as a generalization of time
**Context as a generalization of time**

*Enhance recommendations with contextual information*

- Context is a set of dimensions, or attributes, such as location, companion and time
  - Context attributes may have hierarchical structure

**The big picture:**

- A traditional recommender system considers only two dimensions
  - Users and items
- A context-aware recommender system considers one additional dimension for each context attribute
References

- I. Ntoutsi, K. Stefanidis, K. Norvag and H. P. Kriegel. gRecs: A Group Recommendation System based on User Clustering. DASFAA12
- L. Baltrunas, T. Makcinskas, F. Ricci. Group recommendations with rank aggregation and collaborative filtering. RecSys10
- K. Stefanidis, M. Drosou and E. Pitoura. "You May Also Like" Results in Relational Databases. PersDB09
- M. O’Connor, D. Cosley, J. A. Konstan, J. Riedl. Polylens: A recommender system for groups of user. ECSCW01
- J. S. Breese, D. Heckerman, C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. UAI98