Tutorial: Context In Recommender Systems

Yong Zheng
Center for Web Intelligence
DePaul University, Chicago

Time: 2:30 PM – 6:00 PM, April 4, 2016
Location: Palazzo dei Congressi, Pisa, Italy

The 31st ACM Symposium on Applied Computing, Pisa, Italy, 2016
Introduction

Yong Zheng
Center for Web Intelligence
DePaul University, Chicago, IL, USA
2010 – 2016, PhD in Computer Science, DePaul University
Research: User Modeling and Recommender Systems

Schedule of this Tutorial:
Time: 2:30 PM – 6:00 PM, April 4, 2016
Coffee Break: 4:00 PM – 4:30 PM, April 4, 2016
Topics in this Tutorial

- **Traditional Recommendation**
  e.g., Give me a list of recommended movies to watch

- **Context-aware Recommendation**
  e.g., Give me a list of recommended movies to watch, if
  - Time & Location: at weekend and in cinema
  - Companion: with girlfriend v.s. with Kids

- **Context Suggestion**
  The best time/location to watch movie “Life of Pi”
Outline

• Background: Recommender Systems
  ➢ Introduction and Applications
  ➢ Tasks and Evaluations
  ➢ Traditional Recommendation Algorithms

• Context-aware Recommendation
  ➢ Context Definition, Acquisition and Selection
  ➢ Context Incorporation: Algorithms
  ➢ Other Challenges
  ➢ CARSKit: A Java-Based Open-source RecSys Library

• Context Suggestion

• Summary and Future Directions
Background: RecSys
Outline

• Background: Recommender Systems
  ➢ Introduction and Applications
  ➢ Tasks and Evaluations
  ➢ List of Traditional Recommendation Algorithms
  ➢ Collaborative Filtering
    ▪ User/Item Based Collaborative Filtering
    ▪ Sparse Linear Method
    ▪ Matrix Factorization
Outline

• Background: Recommender Systems
  ➢ Introduction and Applications
  ➢ Tasks and Evaluations
  ➢ List of Traditional Recommendation Algorithms
  ➢ Collaborative Filtering
    ▪ User/Item Based Collaborative Filtering
    ▪ Sparse Linear Method
    ▪ Matrix Factorization
Recommender System (RS)

• RS: item recommendations tailored to user tastes
How it works

✔ yong, choose 3 you like

It will help us find TV shows & movies you’ll love! Click the ones you liked!

Continue
How it works
How it works
How it works

• User Preferences

- Ratings
- Binary Feedback
- Reviews
- Behaviors

Explicit
 Implicit
## Rating-Based Data Sets

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>4</td>
<td></td>
<td>3</td>
<td>4</td>
<td></td>
<td>2</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>U2</td>
<td></td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U3</td>
<td></td>
<td>4</td>
<td></td>
<td>4</td>
<td></td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>U4</td>
<td>3</td>
<td>5</td>
<td></td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U5</td>
<td>2</td>
<td>5</td>
<td></td>
<td>2</td>
<td></td>
<td>4</td>
<td>?</td>
<td>4</td>
</tr>
</tbody>
</table>

User demographic Information: Age, Gender, Race, Country, etc
Item feature information: Movie/Music Genre, Movie director, Music Composer, etc
Outline

• Background: Recommender Systems
  ➢ Introduction and Applications
  ➢ Tasks and Evaluations
  ➢ List of Traditional Recommendation Algorithms
  ➢ Collaborative Filtering
    ▪ User/Item Based Collaborative Filtering
    ▪ Sparse Linear Method
    ▪ Matrix Factorization
## Task and Eval (1): Rating Prediction

**Task: P(U, T) in testing set**

**Prediction error:** \( e = R(U, T) - P(U, T) \)

**Mean Absolute Error (MAE):**
\[
\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |e_i|
\]

**Other evaluation metrics:**
- Root Mean Square Error (RMSE)
- Coverage
- and more ...

### Table: User, Item, Rating

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>T1</td>
<td>4</td>
</tr>
<tr>
<td>U1</td>
<td>T2</td>
<td>3</td>
</tr>
<tr>
<td>U1</td>
<td>T3</td>
<td>3</td>
</tr>
<tr>
<td>U2</td>
<td>T2</td>
<td>4</td>
</tr>
<tr>
<td>U2</td>
<td>T3</td>
<td>5</td>
</tr>
<tr>
<td>U2</td>
<td>T4</td>
<td>5</td>
</tr>
<tr>
<td>U3</td>
<td>T4</td>
<td>4</td>
</tr>
<tr>
<td>U1</td>
<td>T4</td>
<td>3</td>
</tr>
<tr>
<td>U2</td>
<td>T1</td>
<td>2</td>
</tr>
<tr>
<td>U3</td>
<td>T1</td>
<td>3</td>
</tr>
<tr>
<td>U3</td>
<td>T2</td>
<td>3</td>
</tr>
<tr>
<td>U3</td>
<td>T3</td>
<td>4</td>
</tr>
</tbody>
</table>
Task and Eval (1): Rating Prediction

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>T1</td>
<td>4</td>
</tr>
<tr>
<td>U1</td>
<td>T2</td>
<td>3</td>
</tr>
<tr>
<td>U1</td>
<td>T3</td>
<td>3</td>
</tr>
<tr>
<td>U2</td>
<td>T2</td>
<td>4</td>
</tr>
<tr>
<td>U2</td>
<td>T3</td>
<td>5</td>
</tr>
<tr>
<td>U2</td>
<td>T4</td>
<td>5</td>
</tr>
<tr>
<td>U3</td>
<td>T4</td>
<td>4</td>
</tr>
<tr>
<td>U1</td>
<td>T4</td>
<td>3</td>
</tr>
<tr>
<td>U2</td>
<td>T1</td>
<td>2</td>
</tr>
<tr>
<td>U3</td>
<td>T1</td>
<td>3</td>
</tr>
<tr>
<td>U3</td>
<td>T2</td>
<td>3</td>
</tr>
<tr>
<td>U3</td>
<td>T3</td>
<td>4</td>
</tr>
</tbody>
</table>

Task: P(U, T) in testing set

1. Build a model, e.g., P(U, T) = Avg (T)
2. Process of Rating Prediction
   - P(U1, T4) = Avg(T4) = (5+4)/2 = 4.5
   - P(U2, T1) = Avg(T1) = 4/1 = 4
   - P(U3, T1) = Avg(T1) = 4/1 = 4
   - P(U3, T2) = Avg(T2) = (3+4)/2 = 3.5
   - P(U3, T3) = Avg(T3) = (3+5)/2 = 4
3. Evaluation by Metrics
   Mean Absolute Error (MAE) = \( \frac{1}{n} \sum_{i=1}^{n} |e_i| \)
   \( e_i = R(U, T) - P(U, T) \)
   MAE = (|3 - 4.5| + |2 - 4| + |3 - 4| + |3 - 3.5| + |4 - 4|) / 5 = 1
Task and Eval (2): Top-N Recommendation

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>T1</td>
<td>4</td>
</tr>
<tr>
<td>U1</td>
<td>T2</td>
<td>3</td>
</tr>
<tr>
<td>U1</td>
<td>T3</td>
<td>3</td>
</tr>
<tr>
<td>U2</td>
<td>T2</td>
<td>4</td>
</tr>
<tr>
<td>U2</td>
<td>T3</td>
<td>5</td>
</tr>
<tr>
<td>U2</td>
<td>T4</td>
<td>5</td>
</tr>
<tr>
<td>U3</td>
<td>T4</td>
<td>4</td>
</tr>
<tr>
<td>U1</td>
<td>T4</td>
<td>3</td>
</tr>
<tr>
<td>U2</td>
<td>T1</td>
<td>2</td>
</tr>
<tr>
<td>U3</td>
<td>T1</td>
<td>3</td>
</tr>
<tr>
<td>U3</td>
<td>T2</td>
<td>3</td>
</tr>
<tr>
<td>U3</td>
<td>T3</td>
<td>4</td>
</tr>
</tbody>
</table>

Task: Top-N Items to a user U3

Predicted Rank: T3, T1, T4, T2
Real Rank: T3, T2, T1

Then compare the two lists:
Precision@N = # of hits/N

Other evaluation metrics:
- Recall
- Mean Average Precision (MAP)
- Normalized Discounted Cumulative Gain (NDCG)
- Mean Reciprocal Rank (MRR)
- and more ...
Task and Eval (2): Top-N Recommendation

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>T1</td>
<td>4</td>
</tr>
<tr>
<td>U1</td>
<td>T2</td>
<td>3</td>
</tr>
<tr>
<td>U1</td>
<td>T3</td>
<td>3</td>
</tr>
<tr>
<td>U2</td>
<td>T2</td>
<td>4</td>
</tr>
<tr>
<td>U2</td>
<td>T3</td>
<td>5</td>
</tr>
<tr>
<td>U2</td>
<td>T4</td>
<td>5</td>
</tr>
<tr>
<td>U3</td>
<td>T4</td>
<td>4</td>
</tr>
<tr>
<td>U1</td>
<td>T4</td>
<td>3</td>
</tr>
<tr>
<td>U2</td>
<td>T1</td>
<td>2</td>
</tr>
<tr>
<td>U3</td>
<td>T1</td>
<td>3</td>
</tr>
<tr>
<td>U3</td>
<td>T2</td>
<td>3</td>
</tr>
<tr>
<td>U3</td>
<td>T3</td>
<td>4</td>
</tr>
</tbody>
</table>

Train

Task: Top-N Items to user U3

1. Build a model, e.g., $P(U, T) = \text{Avg}(T)$
2. Process of Rating Prediction
   - $P(U3, T1) = \text{Avg}(T1) = 4/1 = 4$
   - $P(U3, T2) = \text{Avg}(T2) = (3+4)/2 = 3.5$
   - $P(U3, T3) = \text{Avg}(T3) = (3+5)/2 = 4$
   - $P(U3, T4) = \text{Avg}(T4) = (4+5)/2 = 3.5$

Predicted Rank: T3, T1, T4, T2
Real Rank: T3, T2, T1

3. Evaluation Based on the two lists
   - Precision@N = # of hits/N
   - Precision@1 = 1/1
   - Precision@2 = 2/2
   - Precision@3 = 2/3

Test
• **Background: Recommender Systems**
  - Introduction and Applications
  - Tasks and Evaluations
  - List of Traditional Recommendation Algorithms
  - Collaborative Filtering
    - User/Item Based Collaborative Filtering
    - Sparse Linear Method
    - Matrix Factorization
Traditional Recommendation Algorithms

- Five Types of algorithms by R. Burke, 2002
  - **Collaborative Filtering**
    e.g., Neighborhood-based algorithms
  - **Content-based Recommender**
    e.g., reusing item features to measure item similarities
  - **Demographic Approaches**
    e.g., reusing user demographic info for marketing purpose
  - **Knowledge-based Algorithms**
    e.g., mining knowledge/relations among users, items
  - **Utility-based Recommender**
    e.g., by maximizing a predefined utility function
Outline

• Background: Recommender Systems
  ➢ Introduction and Applications
  ➢ Tasks and Evaluations
  ➢ List of Traditional Recommendation Algorithms
  ➢ Collaborative Filtering
    ▪ User/Item Based Collaborative Filtering
    ▪ Sparse Linear Method
    ▪ Matrix Factorization
Preliminary: Collaborative Filtering (CF)

• List of three popular CF-based algorithms

- **Neighborhood-based Collaborative Filtering**
  e.g., User/Item based algorithms

- **Sparse Linear Method (SLIM)**
  i.e., a learning-based KNN-based CF approach

- **Matrix Factorization (MF)**
  i.e., a model based collaborative filtering
### User-Based Collaborative Filtering

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U3</td>
<td>4</td>
<td>3</td>
<td>?</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U4</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U5</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **In User-based K-Nearest Neighbor CF (UserKNN)**
  - Assumption: U3’s rating on T5 is similar to other users’ ratings on T5, where these users have similar taste with U3.
  - The “K-Nearest Neighbor” (user neighborhood), can be selected from a list of top-similar users (to U3) identified from the co-ratings by each pair of the users.
User-Based Collaborative Filtering

- UserKNN, P. Resnick, et al., 1994

- User: a; Item: i; User Neighbor: u
- Similarity between user u and a: sim(a, u)
In Item-based K-Nearest Neighbor Neighbor CF (ItemKNN)

- Assumption: U3’s rating on T5 is similar to U3’s rating on similar items.
- The “K-Nearest Neighbor” (item neighborhood), can be selected from a list of top-similar items (to T5) identified from the co-ratings by each pair of the items.
Item-Based Collaborative Filtering

- ItemKNN, B. Sarwar et al., 2001

User: a; Item: i; Item Neighbor: j

Similarity between item i and j: sim(i, j)
Sparse Linear Method

- Sparse Linear Method (SLIM), X. Ning, et al., 2011

\[
\begin{align*}
\text{Rating Prediction in ItemKNN:} & \quad P_{a,i} = \frac{\sum_{j \in N_i} r_{a,j} \times \text{sim}(i, j)}{\sum_{j \in N_i} \text{sim}(i, j)} \\
\text{Score Prediction in SLIM:} & \quad \hat{S}_{i,j} = R_{i,:}W_{:,j} = \sum_{\substack{h=1 \ h \neq j}}^{N} R_{i,h}W_{h,j}
\end{align*}
\]

- Item coefficient (W) is the same as item similarity
- We learn W directly for top-N recommendation Task
Sparse Linear Method

- Sparse Linear Method (SLIM), X. Ning, et al., 2011

\[
\hat{S}_{i,j} = R_{i,:} W_{:,j} = \sum_{h=1}^{N} R_{i,h} W_{h,j}
\]

Minimize

\[
\frac{1}{2} \left\| R_{i,j} - \hat{S}_{i,j} \right\|_F^2 + \frac{\beta_2}{2} \left\| W \right\|_F^2 + \beta_1 \left\| W \right\|_1
\]

Squared Error

L2 Norm (Ridge Reg)

L1 Norm (Lasso Reg)
Matrix Factorization

- Matrix Factorization (MF), Y. Koren, et al., 2009

<table>
<thead>
<tr>
<th>User</th>
<th>HarryPotter</th>
<th>Batman</th>
<th>Spiderman</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>5</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>U2</td>
<td>?</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>U3</td>
<td>4</td>
<td>2</td>
<td>?</td>
</tr>
</tbody>
</table>
Matrix Factorization

- Matrix Factorization (MF), Y. Koren, et al., 2009

R = Rating Matrix, m users, n movies;
P = User Matrix, m users, f latent factors/features;
Q = Item Matrix, n movies, f latent factors/features;

Interpretation:
- $p_u$ indicates how much user likes f latent factors;
- $q_i$ means how much one item obtains f latent factors;
The dot product indicates how much user likes item;
Matrix Factorization

• Matrix Factorization (MF), Y. Koren, et al., 2009

Goal: Try to learn P and Q by minimizing the squared error

\[
\min_{q,p} \sum_{(u,i) \in R} (r_{ui} - x_{ui})^2 + \lambda (|q_i|^2 + |p_u|^2)
\]

Goodness of fit: to reduce the prediction errors;
Regularization term: to alleviate the overfitting;
Matrix Factorization

- Matrix Factorization (MF), Y. Koren, et al., 2009

\[
\min_{q,p} \sum_{(u,i)\in R} (r_{ui} - q_i^T p_u)^2 + \lambda (|q_i|^2 + |p_u|^2)
\]

By using Stochastic Gradient Descent (SGD) or Alternating Least Squares (ALS), we are able to learn the P and Q iteratively.

\[
e_{ui} = r_{ui} - q_i^T p_u
\]

\[
q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)
\]

\[
p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)
\]
Example of Evaluations on CF Algorithms

• Data set: MovieLens-100K

There are 100K ratings given by 943 users on 1,682 movies

Precision@10

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Precision@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemKNN</td>
<td>0.015</td>
</tr>
<tr>
<td>UserKNN</td>
<td>0.025</td>
</tr>
<tr>
<td>MF</td>
<td>0.035</td>
</tr>
<tr>
<td>SLIM</td>
<td>0.035</td>
</tr>
</tbody>
</table>
Summary: Traditional RecSys

• Traditional RecSys: Users $\times$ Items $\rightarrow$ Ratings

• Two recommendation Task:
  - Task 1: Rating Prediction
  - Task 2: Top-N Recommendation

• There are several types of recsys algorithms

• Three popular collaborative filtering (CF):
  - User/Item Based K-Nearest Neighbor CF
  - Sparse Linear Method
  - Matrix Factorization
Context-aware Recommendation
• Context-aware Recommendation
  ➢ Intro: Does context matter?
    ▪ Definition: What is Context?
    ▪ Acquisition: How to collect context?
    ▪ Selection: How to identify the relevant context?
  ➢ Context Incorporation: Algorithms
    ▪ Context Filtering
    ▪ Context Modeling
  ➢ Other Challenges and CARSKit
Outline

• Context-aware Recommendation
  ➢ Intro: Does context matter?
    ▪ Definition: What is Context?
    ▪ Acquisition: How to collect context?
    ▪ Selection: How to identify the relevant context?
  ➢ Context Incorporation: Algorithms
    ▪ Context Filtering
    ▪ Context Modeling
  ➢ Other Challenges and CARSKit
Non-context vs Context

- Decision Making = Rational + Contextual

- Examples:
  - Travel destination: in winter vs in summer
  - Movie watching: with children vs with partner
  - Restaurant: quick lunch vs business dinner
  - Music: workout vs study
What is Context?

• “Context is any information that can be used to characterize the situation of an entity” by Anind K. Dey, 2001

<table>
<thead>
<tr>
<th>How Contextual Factors Change</th>
<th>Knowledge of the RS about the Contextual Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>Fully Observable</td>
</tr>
<tr>
<td>Static</td>
<td>Everything Known about Context</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Context Relevance Is Dynamic</td>
</tr>
</tbody>
</table>

• **Representative Context**: Fully Observable and Static
• **Interactive Context**: Non-fully observable and Dynamic
Interactive Context Adaptation

- Interactive Context: Non-fully observable and Dynamic

List of References:

- N Hariri, B Mobasher, R Burke. "Adapting to user preference changes in interactive recommendation", IJCAI 2015
CARS With Representative Context

• **Observed Context:**
  Contexts are those variables which may change when a same activity is performed again and again.

• **Examples:**
  Watching a movie: time, location, companion, etc
  Listening to a music: time, location, emotions, occasions, etc
  Party or Restaurant: time, location, occasion, etc
  Travels: time, location, weather, transportation condition, etc
Context-aware RecSys (CARS)

- Traditional RS: Users × Items $\rightarrow$ Ratings
- Contextual RS: Users × Items × Contexts $\rightarrow$ Ratings

Example of Multi-dimensional Context-aware Data set

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Rating</th>
<th>Time</th>
<th>Location</th>
<th>Companion</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>T1</td>
<td>3</td>
<td>Weekend</td>
<td>Home</td>
<td>Kids</td>
</tr>
<tr>
<td>U1</td>
<td>T2</td>
<td>5</td>
<td>Weekday</td>
<td>Home</td>
<td>Partner</td>
</tr>
<tr>
<td>U2</td>
<td>T2</td>
<td>2</td>
<td>Weekend</td>
<td>Cinema</td>
<td>Partner</td>
</tr>
<tr>
<td>U2</td>
<td>T3</td>
<td>3</td>
<td>Weekday</td>
<td>Cinema</td>
<td>Family</td>
</tr>
<tr>
<td>U1</td>
<td>T3</td>
<td>?</td>
<td>Weekend</td>
<td>Cinema</td>
<td>Kids</td>
</tr>
</tbody>
</table>
Terminology in CARS

• Example of Multi-dimensional Context-aware Data set

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Rating</th>
<th>Time</th>
<th>Location</th>
<th>Companion</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>T1</td>
<td>3</td>
<td>Weekend</td>
<td>Home</td>
<td>Kids</td>
</tr>
<tr>
<td>U1</td>
<td>T2</td>
<td>5</td>
<td>Weekday</td>
<td>Home</td>
<td>Partner</td>
</tr>
<tr>
<td>U2</td>
<td>T2</td>
<td>2</td>
<td>Weekend</td>
<td>Cinema</td>
<td>Partner</td>
</tr>
<tr>
<td>U2</td>
<td>T3</td>
<td>3</td>
<td>Weekday</td>
<td>Cinema</td>
<td>Family</td>
</tr>
<tr>
<td>U1</td>
<td>T3</td>
<td>?</td>
<td>Weekend</td>
<td>Cinema</td>
<td>Kids</td>
</tr>
</tbody>
</table>

- Context Dimension: time, location, companion
- Context Condition: Weekend/Weekday, Home/Cinema
- Context Situation: {Weekend, Home, Kids}
Context Acquisition

How to Collect the context and user preferences in contexts?

• **By User Surveys or Explicitly Asking for User Inputs**
  Predefine context & ask users to rate items in these situations;
  Or directly ask users about their contexts in user interface;

• **By Usage data**
  The log data usually contains time and location (at least);
  User behaviors can also infer context signals;
Examples: Context Acquisition (RealTime)
Examples: Context Acquisition (Explicit)
Examples: Context Acquisition (Explicit)

![TripAdvisor Review Form](image-url)
Examples: Context Acquisition (Explicit)

Mobile App: South Tyrol Suggests

Personality Collection

Context Collection
Examples: Context Acquisition (PreDefined)
Examples: Context Acquisition (PreDefined)

Google Music: Listen Now

- Working Out
- Cleaning the House
- Hanging Out
- Relaxing at Home
Examples: Context Acquisition (User Behavior)
Context Relevance and Context Selection

Apparently, not all of the context are relevant or influential

• By User Surveys
e.g., which ones are important for you in this domain

• By Feature Selection
e.g., Principal Component Analysis (PCA)
e.g., Linear Discriminant Analysis (LDA)

• By Statistical Analysis or Detection on Contextual Ratings
Statistical test, e.g., Freeman-Halton Test
Other methods: information gain, mutual information, etc

CARS Workshop@ACM RecSys 2012
Context-aware Data Sets

Public Data Set for Research Purpose

- **Food:** AIST Japan Food, Mexico Tijuana Restaurant Data
- **Movies:** AdomMovie, DePaulMovie, LDOS-CoMoDa Data
- **Music:** InCarMusic
- **Travel:** TripAdvisor, South Tyrol Suggests (STS)
- **Mobile:** Frappe

Frappe is a large data set, others are either small or sparse

Downloads and References:

https://github.com/irecsys/CARSKit/tree/master/context-aware_data_sets
• Context-aware Recommendation
  ➢ Intro: Does context matter?
    ▪ Definition: What is Context?
    ▪ Acquisition: How to collect context?
    ▪ Selection: How to identify the relevant context?
  ➢ Context Incorporation: Algorithms
    ▪ Context Filtering
    ▪ Context Modeling
  ➢ Other Challenges and CARSKit
There are three ways to build algorithms for CARS:

(a) Contextual Prefiltering

Data
$U \times I \times C \times R$

Contextualized Data
$U \times I \times R$

2D Recommender
$U \times I \rightarrow R$

Contextual Recommendations
$i_1, i_2, i_3…$

(b) Contextual Postfiltering

Data
$U \times I \times C \times R$

2D Recommender
$U \times I \rightarrow R$

Recommendations
$i_1, i_2, i_3…$

Contextual Recommendations
$i_1, i_2, i_3…$

(c) Contextual Modeling

Data
$U \times I \times C \times R$

MD Recommender
$U \times I \times C \rightarrow R$

Contextual Recommendations
$i_1, i_2, i_3…$
Next, we focus on the following CARS algorithms:

- **Contextual Filtering**: Use Context as Filter
- **Contextual Modeling**: Independent vs Dependent
- **Note**: We focus on context-aware collaborative filtering
Contextual Filtering

- Differential Context Modeling
- UI Splitting
Differential Context Modeling
### Differential Context Modeling

#### Data Sparsity Problem in Contextual Rating

<table>
<thead>
<tr>
<th>User</th>
<th>Movie</th>
<th>Time</th>
<th>Location</th>
<th>Companion</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>Titanic</td>
<td>Weekend</td>
<td>Home</td>
<td>Girlfriend</td>
<td>4</td>
</tr>
<tr>
<td>U2</td>
<td>Titanic</td>
<td>Weekday</td>
<td>Home</td>
<td>Girlfriend</td>
<td>5</td>
</tr>
<tr>
<td>U3</td>
<td>Titanic</td>
<td>Weekday</td>
<td>Cinema</td>
<td>Sister</td>
<td>4</td>
</tr>
<tr>
<td>U1</td>
<td>Titanic</td>
<td>Weekday</td>
<td>Home</td>
<td>Sister</td>
<td>?</td>
</tr>
</tbody>
</table>

**Context Matching** ➔ Only profiles given in <Weekday, Home, Sister>

**Context Relaxation** ➔ Use a subset of context dimensions to match

**Context Weighting** ➔ Use all profiles, but weighted by context similarity
Differential Context Modeling

• Solutions Applied to Collaborative Filtering

**Context Matching** ➔ Only profiles given in <Weekday, Home, Sister>

**Context Relaxation** ➔ Use a subset of context dimensions to match

**Context Weighting** ➔ Use all profiles, but weighted by context similarity

In short, we want to use a subset of rating profiles in collaborative filtering.

There are some research applied such filters to UserKNN or ItemKNN. But there are two main drawbacks:

1). They just apply contexts as filters in one component e.g., the neighborhood selection

2). They just use the same selected contexts as filters i.e., different context dimensions may be useful for different components
Differential Context Modeling

- Context Relaxation

<table>
<thead>
<tr>
<th>User</th>
<th>Movie</th>
<th>Time</th>
<th>Location</th>
<th>Companion</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>Titanic</td>
<td>Weekend</td>
<td>Home</td>
<td>Girlfriend</td>
<td>4</td>
</tr>
<tr>
<td>U2</td>
<td>Titanic</td>
<td>Weekday</td>
<td>Home</td>
<td>Girlfriend</td>
<td>5</td>
</tr>
<tr>
<td>U3</td>
<td>Titanic</td>
<td>Weekday</td>
<td>Cinema</td>
<td>Sister</td>
<td>4</td>
</tr>
<tr>
<td>U1</td>
<td>Titanic</td>
<td>Weekday</td>
<td>Home</td>
<td>Sister</td>
<td>?</td>
</tr>
</tbody>
</table>

Use {Time, Location, Companion} ➔ 0 record matched!
Use {Time, Location} ➔ 1 record matched!
Use {Time} ➔ 2 records matched!

Note: a balance is required for relaxation and accuracy
Differential Context Modeling

• Context Weighting

<table>
<thead>
<tr>
<th>User</th>
<th>Movie</th>
<th>Time</th>
<th>Location</th>
<th>Companion</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>Titanic</td>
<td>Weekend</td>
<td>Home</td>
<td>Girlfriend</td>
<td>4</td>
</tr>
<tr>
<td>U2</td>
<td>Titanic</td>
<td>Weekday</td>
<td>Home</td>
<td>Girlfriend</td>
<td>5</td>
</tr>
<tr>
<td>U3</td>
<td>Titanic</td>
<td>Weekday</td>
<td>Cinema</td>
<td>Sister</td>
<td>4</td>
</tr>
<tr>
<td>U1</td>
<td>Titanic</td>
<td>Weekday</td>
<td>Home</td>
<td>Sister</td>
<td>?</td>
</tr>
</tbody>
</table>

Similarity of contexts is measured by Weighted Jaccard similarity

\[ J(c, d, \sigma) = \frac{\sum_{f \in c \cap d} \sigma_f}{\sum_{f \in c \cup d} \sigma_f} \]

c and d are two contexts. (Two red regions in the Table above.)
\( \sigma \) is the weighting vector \(<w_1, w_2, w_3>\) for three dimensions.
Assume they are equal weights, \( w_1 = w_2 = w_3 = 1 \).
\( J(c, d, \sigma) = \# \text{ of matched dimensions} / \# \text{ of all dimensions} = 2/3 \)
Differential Context Modeling

- Notion of “differential”

In short, we apply different context relaxation and context weighting to each component.
Differential Context Modeling

• **Workflow**

  Step-1: We decompose an algorithm to different components;
  
  Step-2: We try to find optimal context relaxation/weighting:
  
  - In context relaxation, we select optimal context dimensions
  - In context weighting, we find optimal weights for each dimension

• **Optimization Problem**

Assume there are 4 components and 3 context dimensions

<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><strong>DCR</strong></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>DCW</strong></td>
<td>0.2</td>
<td>0.3</td>
<td>0</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
<td>0.1</td>
<td>0.2</td>
<td>0.1</td>
<td>0.5</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Differential Context Modeling

• Optimization Approach
  - Particle Swarm Optimization (PSO)
  - Genetic Algorithms
  - Other non-linear approaches
• How PSO works?

Swarm = a group of birds
Particle = each bird ≈ search entity in algorithm
Vector = bird’s position in the space ≈ Vectors we need in DCR/DCW
Goal = the distance to location of pizza ≈ prediction error

So, how to find goal by swarm intelligence?
1. **Looking for the pizza**
   Assume a machine can tell the distance
2. **Each iteration is an attempt or move**
3. **Cognitive learning from particle itself**
   Am I closer to the pizza comparing with my “best” locations in previous history?
4. **Social Learning from the swarm**
   Hey, my distance is 1 mile. It is the closest!
   Follow me!! Then other birds move towards here.

DCR – Feature selection – Modeled by binary vectors – Binary PSO
DCW – Feature weighting – Modeled by real-number vectors – PSO
Differential Context Modeling

• **Summary**

Pros: Alleviate data sparsity problem in CARS
Cons: Computational complexity in optimization
Cons: Local optimum by non-linear optimizer

Our Suggestion:

- We may just run these optimizations offline to find optimal context relaxation or context weighting solutions; And those optimal solutions can be obtained periodically;
Context-aware Splitting Approaches
The underlying idea in item splitting is that the nature of an item, from the user's point of view, may change in different contextual conditions, hence it may be useful to consider it as two different items. (L. Baltrunas, F. Ricci, RecSys'09) – In short, contexts are dependent with items.
Intro: Item Splitting

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Location</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>M1</td>
<td>Pool</td>
<td>5</td>
</tr>
<tr>
<td>U2</td>
<td>M1</td>
<td>Pool</td>
<td>5</td>
</tr>
<tr>
<td>U3</td>
<td>M1</td>
<td>Pool</td>
<td>5</td>
</tr>
<tr>
<td>U1</td>
<td>M1</td>
<td>Home</td>
<td>2</td>
</tr>
<tr>
<td>U4</td>
<td>M1</td>
<td>Home</td>
<td>3</td>
</tr>
<tr>
<td>U2</td>
<td>M1</td>
<td>Cinema</td>
<td>2</td>
</tr>
</tbody>
</table>

High Rating

Significant difference? Let’s split it !!!

Low Rating

Same movie, different IDs.

M11: being seen at Pool

M12: being seen at Home
**Intro: Item Splitting**

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Loc</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>M1</td>
<td>Pool</td>
<td>5</td>
</tr>
<tr>
<td>U2</td>
<td>M1</td>
<td>Pool</td>
<td>5</td>
</tr>
<tr>
<td>U3</td>
<td>M1</td>
<td>Pool</td>
<td>5</td>
</tr>
<tr>
<td>U1</td>
<td>M1</td>
<td>Home</td>
<td>2</td>
</tr>
<tr>
<td>U4</td>
<td>M1</td>
<td>Home</td>
<td>3</td>
</tr>
<tr>
<td>U2</td>
<td>M1</td>
<td>Cinema</td>
<td>2</td>
</tr>
</tbody>
</table>

Transformation

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>M1</td>
<td>5</td>
</tr>
<tr>
<td>U2</td>
<td>M1</td>
<td>5</td>
</tr>
<tr>
<td>U3</td>
<td>M1</td>
<td>5</td>
</tr>
<tr>
<td>U1</td>
<td>M1</td>
<td>2</td>
</tr>
<tr>
<td>U4</td>
<td>M1</td>
<td>3</td>
</tr>
<tr>
<td>U2</td>
<td>M1</td>
<td>2</td>
</tr>
</tbody>
</table>

**Question:**
How to find such a split? Pool and Non-pool, or Home and Non-home? Which one is the best or optimal split?
• **Splitting Criteria (Impurity Criteria)**

Impurity criteria and significance test are used to make the selection. There are 4 impurity criteria for splitting by L. Baltrunas, et al, RecSys'09: $t_{\text{mean}}$ (t-test), $t_{\text{prop}}$ (z-test), $t_{\text{chi}}$ (chi-square test), $t_{\text{IG}}$ (Information gain)

Take $t_{\text{mean}}$ for example, $t_{\text{mean}}$, is defined using the two-sample t test and computes how significantly different are the means of the rating in the two rating subsets, when the split c (c is a context condition, e.g. location = Pool) is used. The bigger the t value of the test is, the more likely the difference of the means in the two partitions is significant (at 95% confidence value). Choose the largest one!

$$t_{\text{mean}} = \left| \frac{\mu_{i_c} - \mu_{i_{\bar{c}}}}{\sqrt{s_{i_c}/n_{i_c} + s_{i_{\bar{c}}}/n_{i_{\bar{c}}}}} \right|$$
Other Context-aware Splitting Approaches

• **User Splitting and UI Splitting**

  Similarly, the splitting approach can be applied to user too!

  • **User Splitting**: is a similar one. Instead of splitting items, it may be useful to consider one user as two different users, if user demonstrates significantly different preferences across contexts. *(A. Said et al., CARS@RecSys 2011)*  

  *In short, contexts are dependent with users.*

• **UI Splitting**: simply a combination of item splitting and user splitting – both approaches are applied to create a new rating matrix – new users and new items are created in the rating matrix. *(Y. Zheng, et al, ACM SAC 2014)*.  

  *In short, it fuses dependent contexts to users and items simultaneously at the same time.*
Splitting and Transformation

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Rating</th>
<th>Time</th>
<th>Location</th>
<th>Companion</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>T1</td>
<td>3</td>
<td>Weekend</td>
<td>Home</td>
<td>Friend</td>
</tr>
<tr>
<td>U1</td>
<td>T1</td>
<td>5</td>
<td>Weekend</td>
<td>Cinema</td>
<td>Girlfriend</td>
</tr>
<tr>
<td>U1</td>
<td>T1</td>
<td>?</td>
<td>Weekday</td>
<td>Home</td>
<td>Family</td>
</tr>
</tbody>
</table>

(a) by Item Splitting

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>T11</td>
<td>3</td>
</tr>
<tr>
<td>U1</td>
<td>T12</td>
<td>5</td>
</tr>
<tr>
<td>U1</td>
<td>T11</td>
<td>?</td>
</tr>
</tbody>
</table>

(b) by User Splitting

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>U12</td>
<td>T1</td>
<td>3</td>
</tr>
<tr>
<td>U12</td>
<td>T1</td>
<td>5</td>
</tr>
<tr>
<td>U11</td>
<td>T11</td>
<td>?</td>
</tr>
</tbody>
</table>

(c) by UI Splitting

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>U12</td>
<td>T11</td>
<td>3</td>
</tr>
<tr>
<td>U12</td>
<td>T12</td>
<td>5</td>
</tr>
<tr>
<td>U11</td>
<td>T11</td>
<td>?</td>
</tr>
</tbody>
</table>
How Splitting Approaches Work?

• Recommendation Process

Find the best split to perform a splitting approach; After splitting, we obtain a User-item rating matrix; And we can further apply any traditional Recommendation algorithms;

Take Matrix Factorization for example:

Rating Prediction:  
\[ \hat{r}_{ui} = q_i^T p_u \]

Objective function:  
\[ \min_{q^*,p^*} \sum_{(u,i) \in \mathcal{K}} (r_{ui} - q_i^T p_u)^2 + \lambda (\| q_i \|^2 + \| p_u \|^2) \]

Parameter updates Based on SGD

\[ e_{ui} = r_{ui} - q_i^T p_u \]

\[ q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i) \]

\[ p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u) \]
Context-aware Splitting Approaches

• Summary

Pros: Contexts are fused into user and/or item dimensions
Cons: We create new users/items, which increases data sparsity

Our Suggestion:

➢ We may build a hybrid recommender to alleviate the data sparsity or cold-start problems introduced by UI Splitting
Experimental Results

Japan Food Data: 6360 ratings given by 212 users on 20 items within 2 context dimensions
References

- Context-based splitting of item ratings in collaborative filtering. ACM RecSys, 2009
- Alan Said and Ernesto W. De Luca. Inferring Contextual User Profiles – Improving Recommender Performance. CARS@ACM RecSys, 2011
Contextual Modeling

• Independent Contextual Modeling
e.g., Tensor Factorization

• Dependent Contextual Modeling
  1). Deviation-Based Approach
  2). Similarity-Based Approach
Independent Contextual Modeling (Tensor Factorization)
Independent Contextual Modeling

- **Tensor Factorization**

Multi-dimensional space: $Users \times Items \times Contexts \rightarrow Ratings$

Each context variable is modeled as an individual and independent dimension in addition to user & item dims.

Thus we can create a multidimensional space, where rating is the value in the space.
Independent Contextual Modeling

• Tensor Factorization (Optimization)

Multi-dimensional space: Users × Items × Contexts → Ratings

1). By CANDECOMP/PARAFAC (CP) Decomposition
Independent Contextual Modeling

- Tensor Factorization (Optimization)

Multi-dimensional space: Users × Items × Contexts → Ratings

2). By Tucker Decomposition

\[ \mathbf{Y} = \mathbf{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)} + \mathbf{E} \]
Independent Contextual Modeling

- **Tensor Factorization**

  Pros: Straightforward, easily to incorporate contexts into the model

  Cons: 1). Ignore the dependence between contexts and user/item dims
         2). Increased computational cost if more context dimensions

  There are some research working on efficiency improvement on TF, such as reusing GPU computations, and so forth...
Dependent Contextual Modeling

• Dependence between Users/Items and Contexts
  ➢ User and Context, such as user splitting
  ➢ Item and Context, such as item splitting

For example, if a user can be split by time is weekend or not. It tells this user is dependent with this context.

• Dependence between Every two Contexts
  ➢ Deviation-Based: rating deviation between two contexts
  ➢ Similarity-Based: similarity of rating behaviors in two contexts
Deviation-Based Contextual Modeling

- Notion: Contextual Rating Deviation (CRD)

CRD how user’s rating is deviated from context c1 to c2?

<table>
<thead>
<tr>
<th>Context</th>
<th>D1: Time</th>
<th>D2: Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>Weekend</td>
<td>Home</td>
</tr>
<tr>
<td>c2</td>
<td>Weekday</td>
<td>Cinema</td>
</tr>
<tr>
<td>CRD(Di)</td>
<td>0.5</td>
<td>-0.1</td>
</tr>
</tbody>
</table>

CRD(D1) = 0.5 ➔ Users’ rating in Weekday is generally higher than users’ rating at Weekend by 0.5

CRD(D2) = -0.1 ➔ Users’ rating in Cinema is generally lower than users’ rating at Home by 0.1
Deviation-Based Contextual Modeling

- **Notion: Contextual Rating Deviation (CRD)**
  
  CRD how user’s rating is deviated from context $c_1$ to $c_2$?

<table>
<thead>
<tr>
<th>Context</th>
<th>D1: Time</th>
<th>D2: Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>Weekend</td>
<td>Home</td>
</tr>
<tr>
<td>$c_2$</td>
<td>Weekday</td>
<td>Cinema</td>
</tr>
<tr>
<td>CRD($D_i$)</td>
<td>0.5</td>
<td>-0.1</td>
</tr>
</tbody>
</table>

Assume Rating ($U$, $T$, $c_1$) = 4

Predicted Rating ($U$, $T$, $c_2$) = Rating ($U$, $T$, $c_1$) + CRDs

$$= 4 + 0.5 - 0.1 = 4.4$$
Deviation-Based Contextual Modeling

• Build a deviation-based contextual modeling approach

Assume $\emptyset$ is a special situation: without considering context

Assume Rating $(U, T, \emptyset) = \text{Rating } (U, T) = 4$

Predicted Rating $(U, T, c2) = 4 + 0.5 - 0.1 = 4.4$

<table>
<thead>
<tr>
<th>Context</th>
<th>D1: Time</th>
<th>D2: Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>UnKnown</td>
<td>UnKnown</td>
</tr>
<tr>
<td>c2</td>
<td>Weekday</td>
<td>Cinema</td>
</tr>
<tr>
<td>CRD(Di)</td>
<td>0.5</td>
<td>-0.1</td>
</tr>
</tbody>
</table>

In other words, $F(U, T, C) = P(U, T) + \sum_{i=0}^{N} CRD(i)$
Build a deviation-based contextual modeling approach

Simplest model: \( F(U, T, C) = P(U, T) + \sum_{i=0}^{N} CRD(i) \)

User-personalized model: \( F(U, T, C) = P(U, T) + \sum_{i=0}^{N} CRD(i, U) \)

Item-personalized model: \( F(U, T, C) = P(U, T) + \sum_{i=0}^{N} CRD(i, T) \)

Note: \( P(U, T) \) could be a rating prediction by any traditional recommender systems, such as matrix factorization
Deviation-Based Contextual Modeling

- **Context-aware Matrix Factorization (CAMF)**
  
  By Linas Baltrunas, et al., ACM RecSys 2011

  BiasedMF in Traditional RS: \( \hat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i \)

  CAMF_C Approach: \( \hat{r}_{uic_1c_2...c_N} = \mu + b_u + b_i + \sum_{j=1}^{N} CRD(c_j) + p_u^T q_i \)

  CAMF_CU Approach: \( \hat{r}_{uic_1c_2...c_N} = \mu + \sum_{j=1}^{N} CRD(c_j, u) + b_i + p_u^T q_i \)

  CAMF_Ci Approach: \( \hat{r}_{uic_1c_2...c_N} = \mu + b_u + \sum_{j=1}^{N} CRD(c_j, i) + p_u^T q_i \)
Deviation-Based Contextual Modeling

• **Contextual Sparse Linear Method (CSLIM)**

  By Yong Zheng, et al., ACM RecSys 2014

  Rating Prediction in ItemKNN:
  
  \[
P_{a,i} = \frac{\sum_{j \in N_i} r_{a,j} \times \text{sim}(i,j)}{\sum_{j \in N_i} \text{sim}(i,j)}
  \]

  Score Prediction in SLIM:
  
  \[
  \hat{S}_{i,j} = R_{i,:} W_{:,j} = \sum_{h=1}^{N} R_{i,h} W_{h,j}
  \]
Deviation-Based Contextual Modeling

- **Contextual Sparse Linear Method (CSLIM)**

By Yong Zheng, et al., ACM RecSys 2014

SLIM:

\[
\hat{S}_{i,j} = R_{i,:}W_{:,j} = \sum_{\substack{h=1 \\ h \neq i}}^{N} R_{i,h}W_{h,j}
\]

CSLIM_C:

\[
\hat{S}_{i,j,c_1c_2...c_N} = \sum_{\substack{h=1 \\ h \neq j}}^{M} (R_{i,h} + \sum_{k=1}^{N} CRD(k))W_{h,j}
\]

CSLIM_CU:

\[
\hat{S}_{i,j,c_1c_2...c_N} = \sum_{\substack{h=1 \\ h \neq j}}^{M} (R_{i,h} + \sum_{k=1}^{N} CRD(k, i))W_{h,j}
\]

CSLIM_CI:

\[
\hat{S}_{i,j,c_1c_2...c_N} = \sum_{\substack{h=1 \\ h \neq j}}^{M} (R_{i,h} + \sum_{k=1}^{N} CRD(k, h))W_{h,j}
\]
Deviation-Based Contextual Modeling

- Top-10 Recommendation on the Japan Food Data
Similarity-Based Contextual Modeling

• Build a similarity-based contextual modeling approach

Assume $\emptyset$ is a special situation: without considering context

Assume Rating $(U, T, \emptyset) = \text{Rating} (U, T) = 4$

Predicted Rating $(U, T, c2) = 4 \times \text{Sim}(\emptyset, c2)$

In other words, $F(U, T, C) = P(U, T) \times \text{Sim}(\emptyset, C)$

<table>
<thead>
<tr>
<th>Context</th>
<th>D1: Time</th>
<th>D2: Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>UnKnown</td>
<td>UnKnown</td>
</tr>
<tr>
<td>c2</td>
<td>Weekday</td>
<td>Cinema</td>
</tr>
<tr>
<td>Sim(Di)</td>
<td>0.5</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Similitude-Based Contextual Modeling

• Challenge: how to model context similarity, $\text{Sim}(c_1, c_2)$

We propose three representations:
• Independent Context Similarity (ICS)
• Latent Context Similarity (LCS)
• Multidimensional Context Similarity (MCS)
**Similarity-Based Contextual Modeling**

- Sim(c1, c2): Independent Context Similarity (ICS)

\[
\text{Sim}(c1, c2) = \prod_{i=1}^{N} \text{sim}(Di) = 0.5 \times 0.1 = 0.05
\]

**Generally, In ICS:**  
\[
\text{Sim}(c1, c2) = \prod_{i=1}^{N} \text{sim}(Di)
\]
Similarity-Based Contextual Modeling

- **Sim(c1, c2):** Latent Context Similarity (LCS)
  
  In training, we learnt (home, cinema), (work, cinema)
  
  In testing, we need (home, work)

\[ \text{Generally, In LCS: } Sim(c_1, c_2) = \prod_{i=1}^{N} \text{sim}(D_i) \]

\[ \text{Sim}(D_i) = \text{dotProduct} (V_i, V_{i2}) \]

<table>
<thead>
<tr>
<th></th>
<th>f1</th>
<th>f2</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>...</th>
<th>fN</th>
</tr>
</thead>
<tbody>
<tr>
<td>home</td>
<td>0.1</td>
<td>-0.01</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>0.5</td>
</tr>
<tr>
<td>work</td>
<td>0.01</td>
<td>0.2</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>0.01</td>
</tr>
<tr>
<td>cinema</td>
<td>0.3</td>
<td>0.25</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>0.05</td>
</tr>
</tbody>
</table>
Similarity-Based Contextual Modeling

- Sim(c1, c2): Multidimensional Context Similarity (MCS)

In MCS: DisSim(c1, c2) = distance between two points
Similarity-Based Contextual Modeling

- Build algorithms based on traditional recommender

Similarity-Based CAMF:

\[ \hat{r}_{uic_k} = \overrightarrow{p_u} \cdot \overrightarrow{q_i} \cdot \text{Sim}(c_k, c_E) \]

Similarity-Based CSLIM:

\[ \hat{S}_{i,j,c_k} = \sum_{h=1, h\neq j}^{N} R_{i,h} \times W_{h,j} \times \text{Sim}(c_k, c_E) \]

General Similarity-Based CSLIM:

\[ \hat{S}_{i,j,c_k} = \sum_{h=1, h\neq j}^{N} R_{i,h,c_m} \times W_{h,j} \times \text{Sim}(c_k, c_m) \]

In ICS: \( \text{Sim}(c_1, c_2) = \prod_{i=1}^{N} \text{sim}(D_i) \)

In LCS: \( \text{Sim}(c_1, c_2) = \prod_{i=1}^{N} \text{sim}(D_i), \text{sim}(D_i) \text{is dotProduct} \)

In MCS: Dissimilarity is distance, such as Euclidean distance
Deviation-Based Contextual Modeling

- Top-10 Recommendation on the Japan Food Data
Overall Comparison among Best Performers

- Top-10 Recommendation on the Japan Food Data
References

• Context-aware Recommendation

- Intro: Does context matter?
  - Definition: What is Context?
  - Acquisition: How to collect context?
  - Selection: How to identify the relevant context?

- Context Incorporation: Algorithms
  - Context Filtering
  - Context Modeling

- Other Challenges and CARSKit
Other Challenges
Other Challenges

• There could be many other challenges in CARS:

- Numeric v.s. Categorical Context Information
- Cold-start Problems in CARS
- Recommendation Explanations by CARS
- New User Interfaces and Interactions
- New and More Applications
- New Recommendation Opportunities
Other Challenges: Numeric Context

- **List of Categorical Context**
  - Time: morning, evening, weekend, weekday, etc
  - Location: home, cinema, work, party, etc
  - Companion: family, kid, partner, etc

- **How about numeric context**
  - Time: 2016, 6:30 PM, 2 PM to 6 PM (time-aware recsys)
  - Temperature: 12°C, 38°C
  - Principle component by PCA: numeric values
  - Other numeric values in context, how to develop CARS?
Other Challenges: Cold-Start

• Cold-start Problems

Cold-start user: no rating history by this user
Cold-start item: no rating history on this item
Cold-start context: no rating history within this context

• Solution: Hybrid Method by Matthias Braunhofer, et al.

\[
\hat{r}_{uic_1,...,c_k} = (q_i + \sum_{a \in A(i)} x_a)^T p_u + \mu + b_i + b_u + \sum_{t \in T(i)} \sum_{j=1}^{k} b_{tc_j} \\
\hat{r}_{uic_1,...,c_k} = q_i^T (p_u + \sum_{a \in A(u)} y_a) + \mu + b_i + b_u + \sum_{t \in T(i)} \sum_{j=1}^{k} b_{tc_j}
\]
Other Challenges: Explanation

• Recommendation Using social networks (By Netflix)
  The improvement is not significant;
  Unless we explicitly explain it to the end users;

• Recommendation Using context (Open Research)
  Similar thing could happen to context-aware recsys;
  How to use contexts to explain recommendations;
  How to design new user interface to explain;
  How to merge CARS with user-centric evaluations;
Other Challenges: User Interface

- Potential Research Problems in User Interface
  - New UI to collect context;
  - New UI to interact with users friendly and smoothly;
  - New UI to explain context-aware recommendation;
  - New UI to avoid debates on user privacy;
  - User privacy problems in context collection & usage
Other Challenges: New Applications

• More applications are in demand:
  
  ➢ Not only e-commerce, movies, music, etc
  ➢ Tourism: Trip planner, Traffic analyzer and planner
  ➢ MOOC: online learning via different characteristics
  ➢ Life Long: Digital health, daily activity tracker
  ➢ Shared Economy: Uber, Airbnb
Other Challenges: New Opportunity

• CARS enable new recommendation opportunities

Context Suggestion

We will introduce later in this tutorial.
CARSKit: Recommendation Library
Motivations to Build a Recommendation Library

1). Standard Implementations for popular algorithms
2). Standard platform for benchmark or evaluations
3). Helpful for both research purpose and industry practice
4). Helpful as tools in teaching and learning
There are many recommendation library for traditional recommendation.

**Users × Items → Ratings**

<table>
<thead>
<tr>
<th></th>
<th>Mahout</th>
<th>GraphLab Create</th>
<th>MyMediaLite</th>
<th>EasyRec</th>
<th>LensKit</th>
<th>LibRec</th>
<th>CARSKit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Latest release (date)</strong></td>
<td>05/2015</td>
<td>2015</td>
<td>09/2013</td>
<td>05/2013</td>
<td>05/2015</td>
<td>06/2015</td>
<td>09/2015</td>
</tr>
<tr>
<td><strong>Latest version</strong></td>
<td>0.10.1</td>
<td>1.4.1</td>
<td>3.10</td>
<td>0.98</td>
<td>2.2</td>
<td>1.3.2</td>
<td>0.1.0</td>
</tr>
<tr>
<td><strong>Active updates</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>License</strong></td>
<td>Apache</td>
<td>BSD, AGPL</td>
<td>GPL</td>
<td>N/A</td>
<td>GPL</td>
<td>GPL</td>
<td>GPL</td>
</tr>
<tr>
<td><strong>Programming language</strong></td>
<td>Java</td>
<td>Python, C++</td>
<td>C#</td>
<td>WebService</td>
<td>Java</td>
<td>Java</td>
<td>Java</td>
</tr>
<tr>
<td><strong>Running platform</strong></td>
<td>JVM</td>
<td>Any</td>
<td>.NET</td>
<td>WebApp</td>
<td>JVM</td>
<td>JVM</td>
<td>JVM</td>
</tr>
<tr>
<td><strong>Command line</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Documentation</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Online updates</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Algorithm diversity</strong></td>
<td>Classical</td>
<td>Classical</td>
<td>State-of-the-art</td>
<td>Classical</td>
<td>Classical</td>
<td>State-of-the-art</td>
<td>CARS</td>
</tr>
<tr>
<td><strong>Distributed computing</strong></td>
<td>Partial</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Open source</strong></td>
<td>Yes</td>
<td>Partial</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
CARSKit: A Java-based Open-source Context-aware Recommendation Library

CARSKit: https://github.com/irecsys/CARSKit

Users × Items × Contexts → Ratings

1. Download the JAR library, i.e., CARSKit.jar
2. Prepare your data
   - userid, itemid, rating, Time, Location, Companion
   - 1003, tt0133093, 1, Weekday, Cinema, Alone
   - 1067, tt1099212, 2, Weekend, Home, Partner
   - 1003, tt1099212, 1, Weekday, Cinema, Alone
   - 1003, tt0454876, 1, Weekday, Home, Family
3. Setting: setting.conf
   ```plaintext
   dataset.ratings.wins=E:\data\DePaulMovie\ratings.txt
   dataset.ratings.lins=/users/yzheng/data/DePaulMovie/ratings.txt
   ratings.setup=-threshold -1 -datatransformation 1
   recommender=CAMF_C
   evaluation.setup=cv -k 5 -p on --rand-seed 1 --test-view all --early-stop RMSE
   item.ranking=off -topN 10
   num.factors=10
   num.max.iter=100
   learn.rate=2e-4 -max -1 -bold-driver
   reg.lambda=0.001 -u 0.001 -i 0.001 -b 0.001 -s 0.001 -c 0.001
   ```
4. Run: java –jar CARSKit.jar –c setting.conf
Sample of Outputs: Data Statistics

```plaintext
Dataset: E:\Experiment\Data\depaulmovie\CARSKit.Workspace\ratings_binary.txt
User amount: 97
Item amount: 79
Rate amount: 5035
Context dimensions: 3 (companion, location, time)
Context conditions: 10 (companion: 4, location: 3, time: 3)
Context situations: 13
Contextual Data density: 21.8757%
Scale distribution: [2.0 x 625, 4.0 x 1209, 1.0 x 829, 5.0 x 1367, 3.0 x 1005]

Average value of all ratings: 3.329688
Standard deviation of all ratings: 1.414732
Mode of all rating values: 5.000000
Median of all rating values: 4.000000
```
Sample of Outputs:

1). Results by Rating Prediction Task

Final Results by CAMF_C, **MAE**: 0.714544, **RMSE**: 0.960389, **NAME**: 0.178636, **rMAE**: 0.683435, **rRMSE**: 1.002392, **MPE**: 0.000000, numFactors: 10, numIter: 100, lrate: 2.0E-4, maxlrate: -1.0, regB: 0.001, regU: 0.001, regI: 0.001, regC: 0.001, isBoldDriver: true, Time: '00:01','00:00'

2). Results by Top-N Recommendation Task

Final Results by CAMF_C, **Pre5**: 0.048756, **Pre10**: 0.050576, **Rec5**: 0.094997, **Rec10**: 0.190364, **AUC**: 0.653558, **MAP**: 0.054762, **NDCG**: 0.105859, **MRR**: 0.107495, numFactors: 10, numIter: 100, lrate: 2.0E-4, maxlrate: -1.0, regB: 0.001, regU: 0.001, regI: 0.001, regC: 0.001, isBoldDriver: true, Time: '00:01','00:00'
Outline

• Background: Recommender Systems
  ➢ Introduction and Applications
  ➢ Tasks and Evaluations
  ➢ Traditional Recommendation Algorithms
• Context-aware Recommendation
  ➢ Context Definition, Acquisition and Selection
  ➢ Context Incorporation: Algorithms
  ➢ Other Challenges
  ➢ CARSKit: A Java-Based Open-source RecSys Library
• Context Suggestion
• Summary and Future Directions
Context Suggestion
Context Suggestion

- Task: Suggest a list of contexts to users (on items)
• Motivation-1: Maximize user experience

User Experience (UX) refers to a person's emotions and attitudes about using a particular product, system or service.
Context Suggestion: Motivations

• To maximize user experience (UX)

Example: Evolution in Retail
To maximize user experience (UX)

Example: Evolution in Retails
Context Suggestion: Motivations

- To maximize user experience (UX)
  
  Example: Evolution in Retails

![Diagram showing the context, service, and product with icons representing different contexts and services.]

**Context**

**Service**

**Product**
Motivation-1: Maximize user experience

It is not enough to recommend items only
Context Suggestion: Motivations

- Motivation-2: Contribute to Context Collection
  Predefine contexts and suggest them to users
Context Suggestion: Motivations

- Motivation-3: Connect with Context-aware RecSys

User’s actions on context is a context-query to system
• There could be many potential applications:

<table>
<thead>
<tr>
<th>Context Recommendation</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Suggestion</td>
<td>user</td>
<td>a list of contexts</td>
</tr>
<tr>
<td></td>
<td>item</td>
<td>a list of contexts</td>
</tr>
<tr>
<td></td>
<td>user, item</td>
<td>a list of contexts</td>
</tr>
<tr>
<td>Context Suggestion</td>
<td>user</td>
<td>items + contexts</td>
</tr>
<tr>
<td>As Explanations</td>
<td>item</td>
<td>users + contexts</td>
</tr>
<tr>
<td>Bundle Suggestion</td>
<td>user, item</td>
<td>contexts + items</td>
</tr>
<tr>
<td></td>
<td>user, item</td>
<td>contexts + users</td>
</tr>
</tbody>
</table>
Context Suggestion: Applications

- There could be many potential applications:

<table>
<thead>
<tr>
<th>Context Recommendation</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Suggestion</td>
<td>user</td>
<td>a list of contexts</td>
</tr>
<tr>
<td></td>
<td>item</td>
<td>a list of contexts</td>
</tr>
<tr>
<td></td>
<td>user, item</td>
<td>a list of contexts</td>
</tr>
<tr>
<td>Context Suggestion As Explanations</td>
<td>user</td>
<td>items + contexts</td>
</tr>
<tr>
<td></td>
<td>item</td>
<td>users + contexts</td>
</tr>
<tr>
<td>Bundle Suggestion</td>
<td>user, item</td>
<td>contexts + items</td>
</tr>
<tr>
<td></td>
<td>user, item</td>
<td>contexts + users</td>
</tr>
</tbody>
</table>
Context Suggestion: Applications

Real Examples: Google Music

- Working Out
- Today's Biggest Hits
- Family Time
- Happy Hour

Recent activity
Recently played or added

- Coldplay Radio
- Wanting Radio
- Astropilot Radio
- Avril Lavigne Radio

Input is a single user
Context Suggestion: Applications

Vedge: 100 Plates Large and Small That Redefine Vegetable Cooking

by Rich Landau (Author), Kate Jacoby (Author), Joe Yonan (Foreword)

54 customer reviews

This is a good gift for your Mom!

See all formats and editions

<table>
<thead>
<tr>
<th>Format</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kindle</td>
<td>$9.48</td>
</tr>
<tr>
<td>Hardcover</td>
<td>$18.27</td>
</tr>
<tr>
<td>Paperback</td>
<td>from $32.67</td>
</tr>
</tbody>
</table>

Read with our free app

- 28 Used from $12.07
- 48 New from $15.00
- 2 Used from $32.67
- 5 New from $34.00

The most exciting vegetable cooking in the nation is happening at Vedge, where in an elegant nineteenth-century townhouse in Philadelphia, chef-proprietors Rich Landau and Kate Jacoby serve exceptionally flavorful fare that is wowing vegans, vegetarians, and carnivores alike.

Now, Landau and Jacoby share their passion for ingenious vegetable cooking. The more than 100 recipes here—such as Fingerling Potatoes with Creamy Worcestershire Sauce, Pho with Roasted Butternut Squash, Seared French Beans with Caper Bagna Cauda, and Eggplant
Context Suggestion: Applications

- There could be many potential applications:

<table>
<thead>
<tr>
<th>Context Recommendation</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Suggestion</td>
<td>user</td>
<td>a list of contexts</td>
</tr>
<tr>
<td></td>
<td>item</td>
<td>a list of contexts</td>
</tr>
<tr>
<td></td>
<td>user, item</td>
<td>a list of contexts</td>
</tr>
<tr>
<td>Context Suggestion</td>
<td>user</td>
<td>items + contexts</td>
</tr>
<tr>
<td>As Explanations</td>
<td>item</td>
<td>users + contexts</td>
</tr>
<tr>
<td>Bundle Suggestion</td>
<td>user, item</td>
<td>contexts + items</td>
</tr>
<tr>
<td></td>
<td>user, item</td>
<td>contexts + users</td>
</tr>
</tbody>
</table>
Context Suggestion: Applications

Input is user;
Output is kid + movies

Popular

Madagascar
King Julien
The Clone Wars
Phineas and Ferb
My Little Pony
Context Suggestion: Applications

As a gift for Mother’s Day

Input is user; Output is day + books

Customers Who Bought This Item Also Bought

- **Magical Journey: An Apprenticeship in Contentment**
  - Author: Katrina Kenison
  - Rating: 4.5 stars
  - Format: Paperback
  - Price: $12.05 with Prime

- **Mitten Strings for God: Reflections for Mothers in a Hurry**
  - Author: Katrina Kenison
  - Rating: 4.9 stars
  - Format: Paperback
  - Price: $11.95 with Prime

- **Life Is a Verb: 37 Days To Wake Up, Be Mindful, And Live Intentionally**
  - Author: Patti Digh
  - Rating: 4.7 stars
  - Format: Paperback
  - Price: $13.96 with Prime

- **Lift**
  - Author: Kelly Corrigan
  - Rating: 4.5 stars
  - Format: Hardcover
  - Price: $13.37 with Prime
## Context Suggestion: Applications

- There could be many potential applications:

<table>
<thead>
<tr>
<th>Context Recommendation</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Context Suggestion</strong></td>
<td>user</td>
<td>a list of contexts</td>
</tr>
<tr>
<td></td>
<td>item</td>
<td>a list of contexts</td>
</tr>
<tr>
<td></td>
<td>user, item</td>
<td>a list of contexts</td>
</tr>
<tr>
<td><strong>Context Suggestion As Explanations</strong></td>
<td>user</td>
<td>items + contexts</td>
</tr>
<tr>
<td></td>
<td>item</td>
<td>users + contexts</td>
</tr>
<tr>
<td><strong>Bundle Suggestion</strong></td>
<td>user, item</td>
<td>contexts + items</td>
</tr>
<tr>
<td></td>
<td>user, item</td>
<td>contexts + users</td>
</tr>
</tbody>
</table>
Context Suggestion: Applications

A list of item recommendation associated with context
Context Suggestion

• Task: Suggest a list of appropriate contexts to users
  For example: Where should I watch movie Life Of Pi

• Timeline
  In 2008, proposed by Tutorial at ACM RecSys 2008
  In 2010, first attempt by Linas et al. ACM RecSys 2010
  In 2014, formal discussion by Yong et al., IEEE/ACM WI 2014
  In 2015, proposal more applications by Yong, IEEE ICDM 2015
  In 2016, working on new solutions for related problems
• There could be many applications, we focus on two tasks

1). UI-Oriented Context Suggestion
Task: suggest contexts to <user, item>
Example: time & location to watch movie *Life of Pi*

2). User-Oriented Context Suggestion
Task: suggest contexts to each user
Example: Google Music, Pandora, Youtube, etc
UI-Oriented Context Suggestion

Solution 1). Multilabel classification (MLC)

- KNN classifiers by Linas et al., ACM RecSys 2010
- Other MLC by Zheng et al., IEEE/ACM WI, 2014

1). Binary Classification
Question: Is this an apple? Yes or No.

2). Multi-class Classification
Question: Is this an apple, banana or orange?

3). Multi-label Classification
Use appropriate words to describe it:
Red, Apple, Fruit, Tech, Mac, iPhone

Color, Shape, Weight, Origin, Taste, Price, Vitamin c

In our case, we use user and item information as inputs and features to learn label predictions
UI-Oriented Context Suggestion

Solution 2). Context-aware Recommendation

We can reuse CARS algorithms to recommend contexts. For example, Tensor Factorization

- We put all conditions into a single dimension: context
- Then we create 3D space: user, item, context
- We recommend contexts for each <user, item>

In other ways, we view contexts as tags to be recommended; other CARS algorithms can also be applied
User-Oriented Context Suggestion

It can be viewed as a process of context acquisition. But recommendation task is still involved in it.
User-Oriented Context Suggestion

There could be several potential solutions:

1). Most popular or user-popular context suggestion;
2). Most recent or user-recent context suggestion;
3). Collaborative suggestion based on other users’ tastes;
4). Reuse context-aware recommendation algorithms;
Context Suggestion: Challenges

It is still a novel and emerging research direction. There are several challenges to be solved:

1). Evaluations
We do not have user’s taste on context

2). Solutions
Is personalized required? Any personalized solutions? Popular suggestion is a good solution?

3). User Interface
How to build appropriate UI to interact with users
References

- L Baltrunas, M Kaminskas, F Ricci, et al. Best usage context prediction for music tracks. CARS@ACM RecSys, 2010
Outline

• Background: Recommender Systems
  ➢ Introduction and Applications
  ➢ Tasks and Evaluations
  ➢ Traditional Recommendation Algorithms

• Context-aware Recommendation
  ➢ Context Definition, Acquisition and Selection
  ➢ Context Incorporation: Algorithms
  ➢ Other Challenges
  ➢ CARSKit: A Java-Based Open-source RecSys Library

• Context Suggestion

• Summary and Future Directions
Topics in this Tutorial

• Traditional Recommendation
  e.g., Give me a list of recommended movies to watch

• Context-aware Recommendation
  e.g., Give me a list of recommended movies to watch, if
  ➢ Time & Location: at weekend and in cinema
  ➢ Companion: with girlfriend v.s. with Kids

• Context Suggestion
  The best time/location to watch movie “Life of Pi”
Details in this Tutorial

• Background: Recommender Systems
  ➢ Introduction and Applications
  ➢ Tasks and Evaluations
  ➢ Traditional Recommendation Algorithms

• Context-aware Recommendation
  ➢ Context Definition, Acquisition and Selection
  ➢ Context Incorporation: Algorithms
  ➢ Other Challenges
  ➢ CARSKit: A Java-Based Open-source RecSys Library

• Context Suggestion: App, Solution and Challenges
Future Research

• Context-aware Recommendation
  - Treat Numeric Context Information
  - Cold-start Problems in CARS
  - Recommendation Explanation by Context
  - User Interface and More applications by CARS

• Context Suggestion
  - Data collection for evaluations
  - Examine different algorithms on real-world data
  - Design new user interface and applications
List of Tutorials/Keynotes about CARS

- Bamshad Mobasher, “Contextual User Modeling for Recommendation”, In CARS Workshop@ACM RecSys, 2010
- Francesco Ricci, “Contextualizing Recommendations”, In CARS Workshop@ACM RecSys, 2012
Tutorial: Context In Recommender Systems

Yong Zheng
Center for Web Intelligence
DePaul University, Chicago

Thanks

The 31st ACM Symposium on Applied Computing, Pisa, Italy, 2016