A roadmap for privacy preserving tourist recommendation system

Alan Wecker[1,*], Noa Tuval[1], Alain Hertz[2], Muhammad Mahamid[1] and Tsvi Kuflik[1]

[1] *University of Haifa, Haifa, Israel*

[2] *Polytechnique Montreal, Montreal Canada*

Agenda

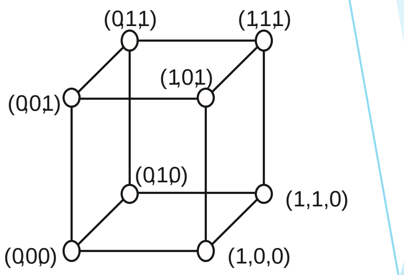
- Motivation
- System
  - User Interface
  - Hypercube Recommendation Engine
- Advantages and Challenges
- **Practical considerations to work on mobile device**
- **Evaluation**

Motivation

- Preserving User Privacy
  - Risks
    - Privacy invaded, Being targeted by service providers, …
    - Alternate opinion: User doesn’t really care since they may be seen in restaurant which is public place
      - Different type of exposure
  - Solutions
    - Anonymity
      - But: Not 100%, BZIP, etc…
    - Content Based Recommendations
      - Theoretically do not require to share any user information

Theoretical / practical solution

- Content-based recommender system
  - Item representation
    - Hypercube architecture
  - User model
    - Reasoning on user rating given to items represented as binary vectors in the hypercube



Definitions

Let U be a set of users

Let I be a set of items

Let A be a set of Boolean attributes.

Let $v(i, a)$ be the value of attribute a for item i.

A vector $\boldsymbol{x}_i$ can be associated with every item i so that the jth component of $\boldsymbol{x}_i$ is equal to 1 if and only if $v(i, a)$ is true, where a is the jth attribute

Example

I is a set of restaurants

1st attribute : low cost 2nd attribute : offer vegetarian food 3rd attribute: with a terrace facing the sea

The vector (0,1,1) is associated with an expensive restaurant, facing the sea, where vegetarians can eat.

Every user $u \in U$ has preferences, and we can therefore also associate a vector $\boldsymbol{y}_u$ to u so that the jth component of $\boldsymbol{y}_u$ is equal to 1 if and only if u has interest for the jth attribute.

Example

If a user u likes vegetarian low price restaurants, even if they have no terrace facing the sea, then $\boldsymbol{y}_u = (1{,}1{,}0)$.

The **Hamming distance** (number of different components) between $\boldsymbol{y}_u = (1{,}1{,}0)$ and $\boldsymbol{x}_i = (0{,}1{,}1)$ (which correspond to an expensive vegetarian restaurant facing the sea) is 2.

$$d(\boldsymbol{x}_i, \boldsymbol{y}_u) = \mathrm{d}((0{,}11), (1{,}1{,}0)) = 2$$

Item representation

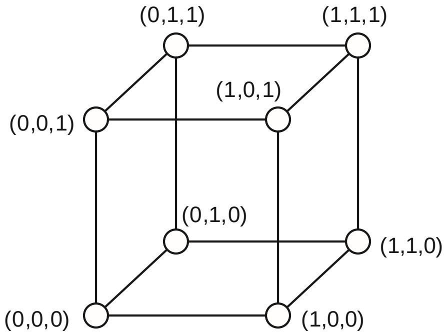
Let A be an ordered set of *n* Boolean attributes.

Let Q_n be the *n*-dimensional hypercube with vertex set $\{0, 1\}^n$, and where two vertices are linked with an edge if and only if their differ in exactly one component.

The items of the recommender system are vertices in Q_n

$\Rightarrow$ an item *i* is associated with a vertex $\mathbf{v}^i = (v^i_1 ,\ldots, v^i_n)$ where $v^i_j = 1$ if item *i* has attribute *j*, $v^i_j = 0$ otherwise.

Note that two items with the same attributes are associated with the same vertex in Q_n.
We can therefore consider every vertex of Q_n as an item type.



User representation

The preferences of a user *u* of the recommender system are modeled as a vector $\mathbf{w}^u = (w^u_1, \ldots, w^u_n)$ in $\{-1,0,1\}^n$ where

- $w^u_j = 1$ if u likes attribute *j*
- $w^u_j = 0$ if u does not care about attribute *j*
- $w^u_j = -1$ if u does not like attribute *j*.

We define $\mathbf{w}^u$ as the user profile

Approximation of the user profile

The distance d($\mathbf{v}^i$, $\mathbf{w}^u$) between an item *i* and the profile of user *u* is the number of components *j* (i.e., attributes) such that

- either $v^i_j = 1$ and $w^u_j = -1$ (i.e., the item *i* has attribute *j* that the user does not like
- or $v^i_j = 0$ and $w^u_j = 1$ (i.e., the item *i* does not have attribute *j* that the user likes).

A rating according to an *s*-star scale (i.e., a rating in {1, 2, …., *s*) can be transformed into a distance to the user preferences.

- If a user gives *s* stars to an item *i*, it means that he likes all attributes in *i*, and dislikes all others.
- If a user gives 1 star to an item *i*, it means that he does not like the attributes in *i*, and likes all others.

We transform a rating r_i in {1,2,…,*s*} into a number δ_i of attributes that do not fit with the user preference.

$$\delta_i = \tau(r_i) = n - \frac{n(r_i - 1)}{s - 1}$$

$$r_i = 1 \Rightarrow \delta_i = n$$
$$r_i = s \Rightarrow \delta_i = 0$$

System

- Server
  - Collect generic restaurant information
  - Generate Hypercube
- User Device
  - User Interface
  - Search hyper-cube according to user preferences

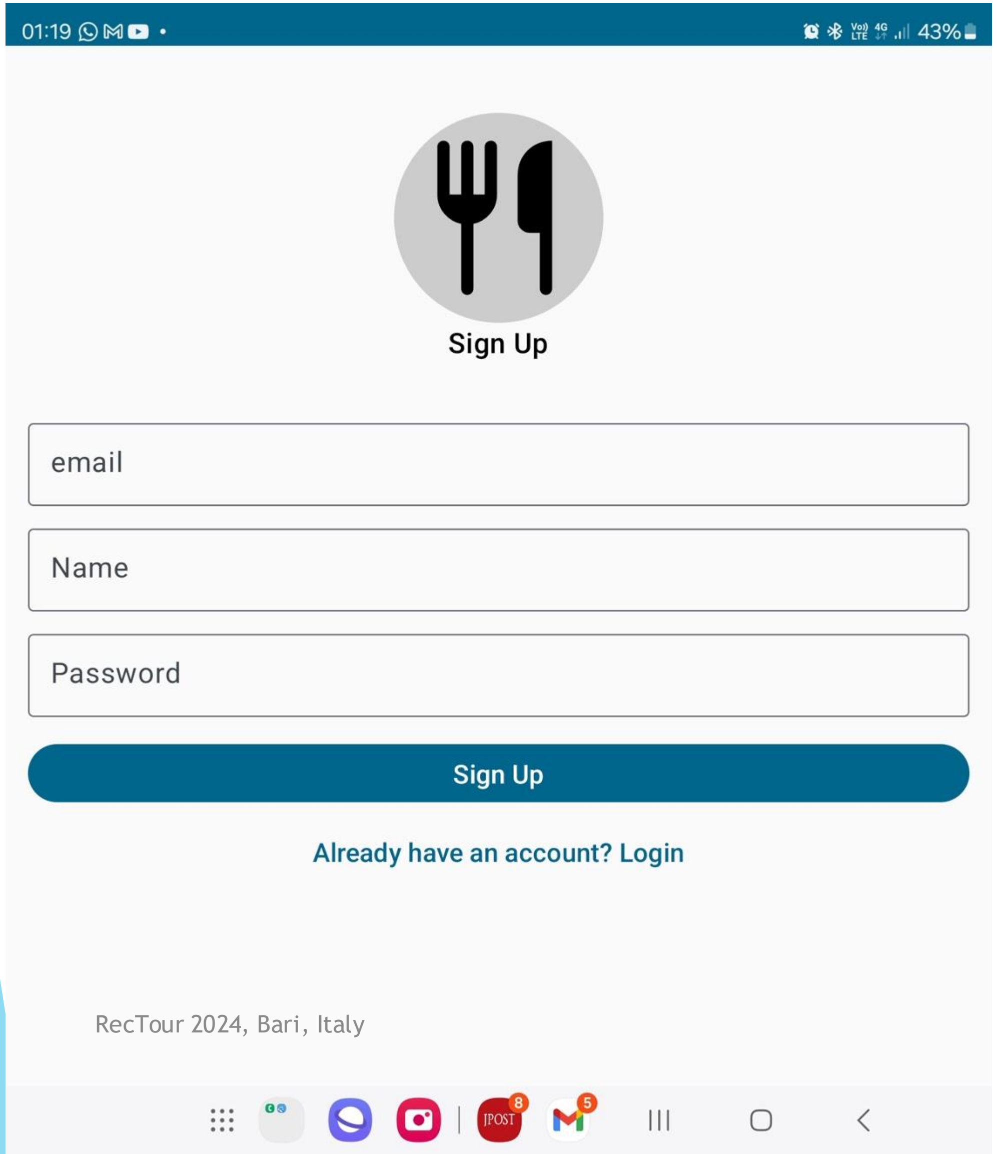
- JSON file
  - Descriptive information
  - Restaurant Features
    - Translate multi-valued criteria in binary
      - Thus Cuisine gets translated into:

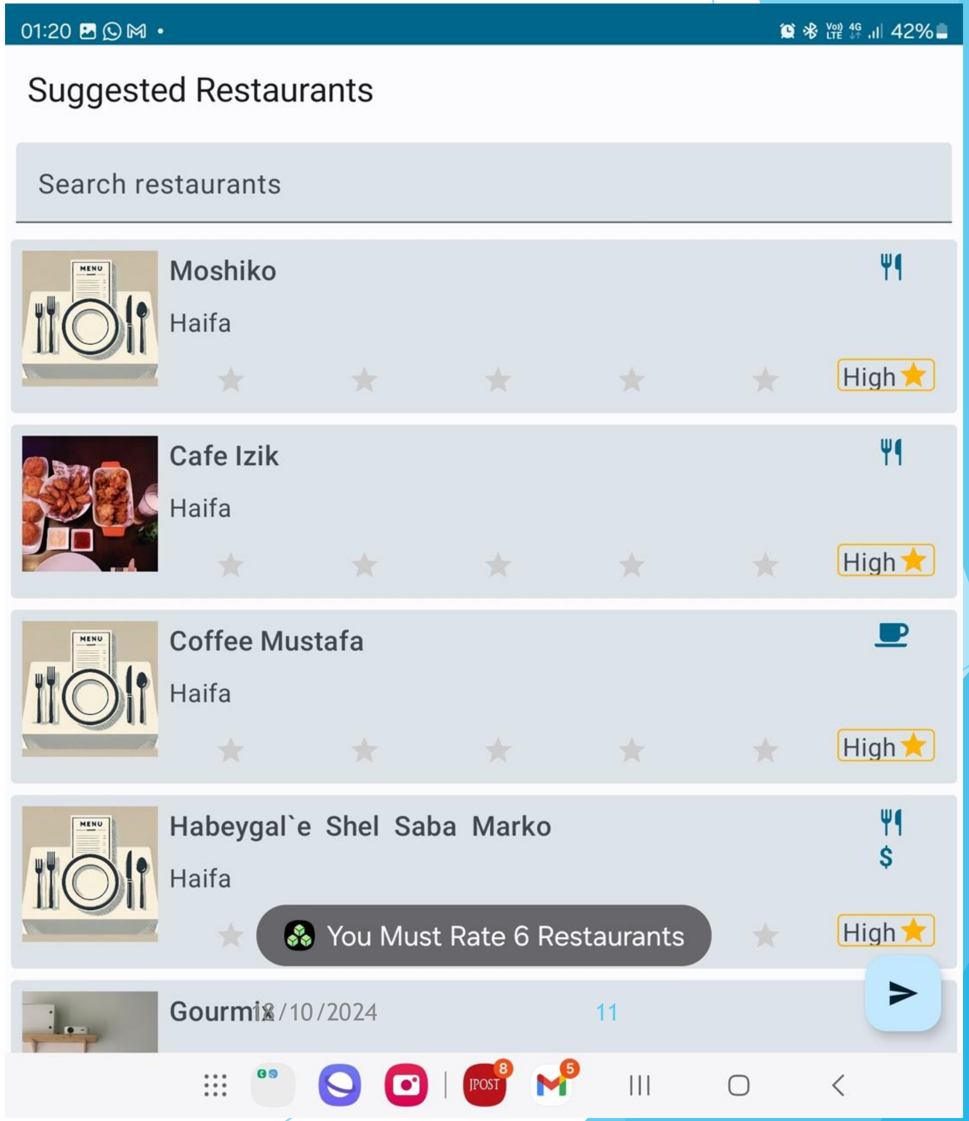
        IsChinese

        IsTurkish

        IsFastFood

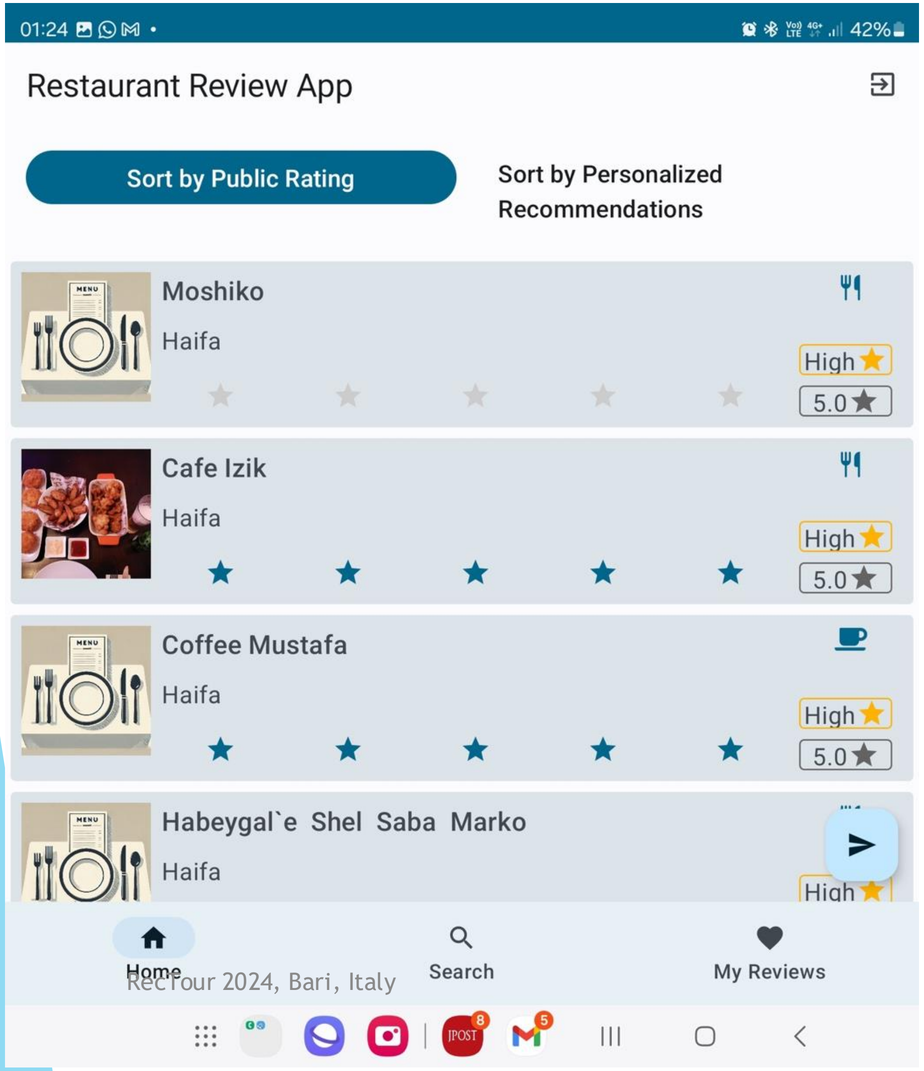
User Interface I (Initialization)





You Must Rate 6 Restaurants

User Interface II



01:24 · 42%

Restaurant Review App

Sort by Public Rating

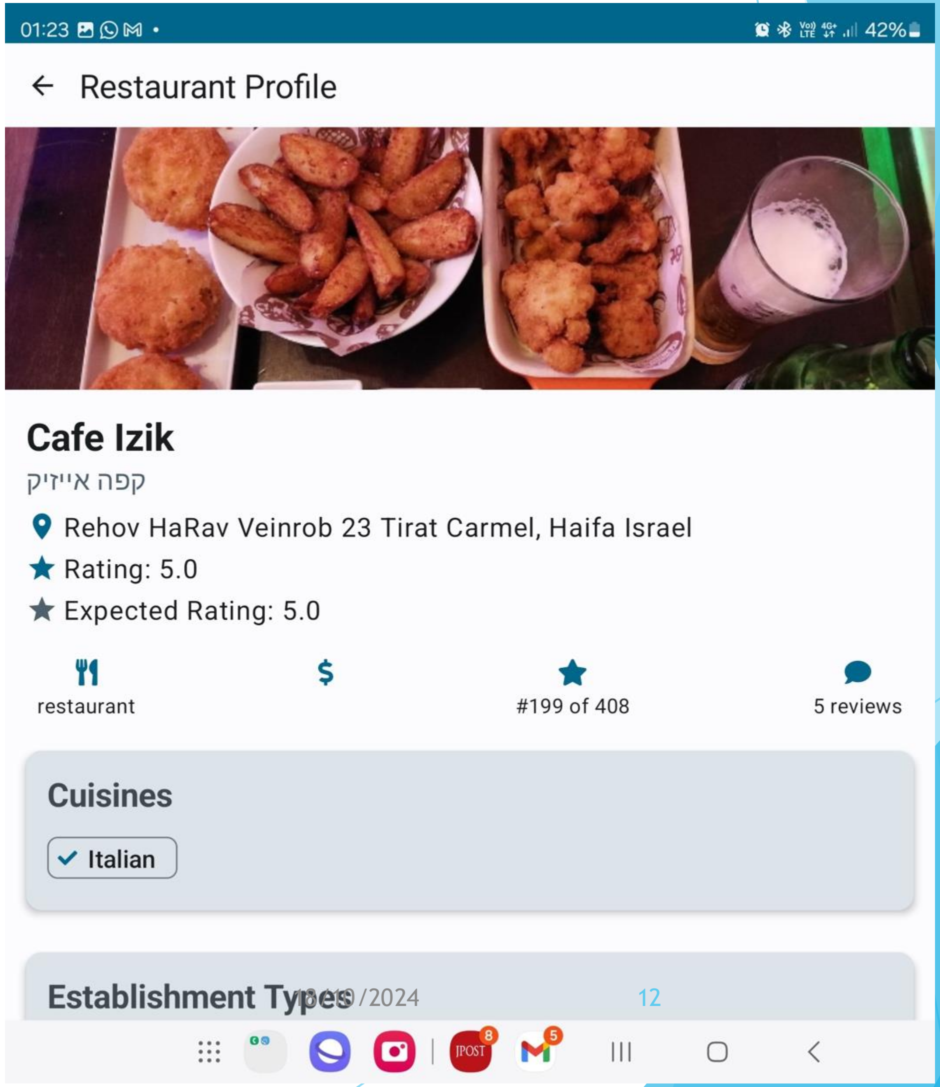
Sort by Personalized Recommendations

Moshiko
Haifa
High ★
5.0★

Cafe Izik
Haifa
★ ★ ★ ★ ★
High ★
5.0★

Coffee Mustafa
Haifa
★ ★ ★ ★ ★
High ★
5.0★

Habeygal`e Shel Saba Marko
Haifa
High ★



01:23 · 42%

Restaurant Profile

Cafe Izik
קפה אייזיק
Rehov HaRav Veinrob 23 Tirat Carmel, Haifa Israel
★ Rating: 5.0
★ Expected Rating: 5.0

Cuisines
✓ Italian

Establishment Types

Practical Considerations

- Integer Linear Program on mobile
  - Find package
  - Port GLPK
  - Program subset needed
- Feature Reduction
  - Remove unique features (globally)
  - Remove features not contained in all items user ranked

Pros and Cons

Advantages

- Can model complex feature sets
- Need only a small number of user ratings
- Does not need to share user data

Challenges

- What is financial advantage to service provider
  - Usually personalized ads
- How can we share data among multiple personal devices
- Cold Start problem

Evaluation

- Goal: See if system gives:
  - 1) reasonable (i.e. similar user satisfaction) while maintaining privacy and
  - 2) reasonable (fast) response

- Method (for 1): Compare to another recommender system's ranking
- Method: (for 2): Measure speed, measure algorithmic complexity

- Remember goal is provide reasonable ranking (measured by user satisfaction), not necessarily the most precise or complete.

Thank you!

Questions?