Learning to Recommend with Location and Context

Chen Cheng

Supervisors: Michael R. Lyu and Irwin King

The Chinese University of Hong Kong

August 31, 2015

Outline

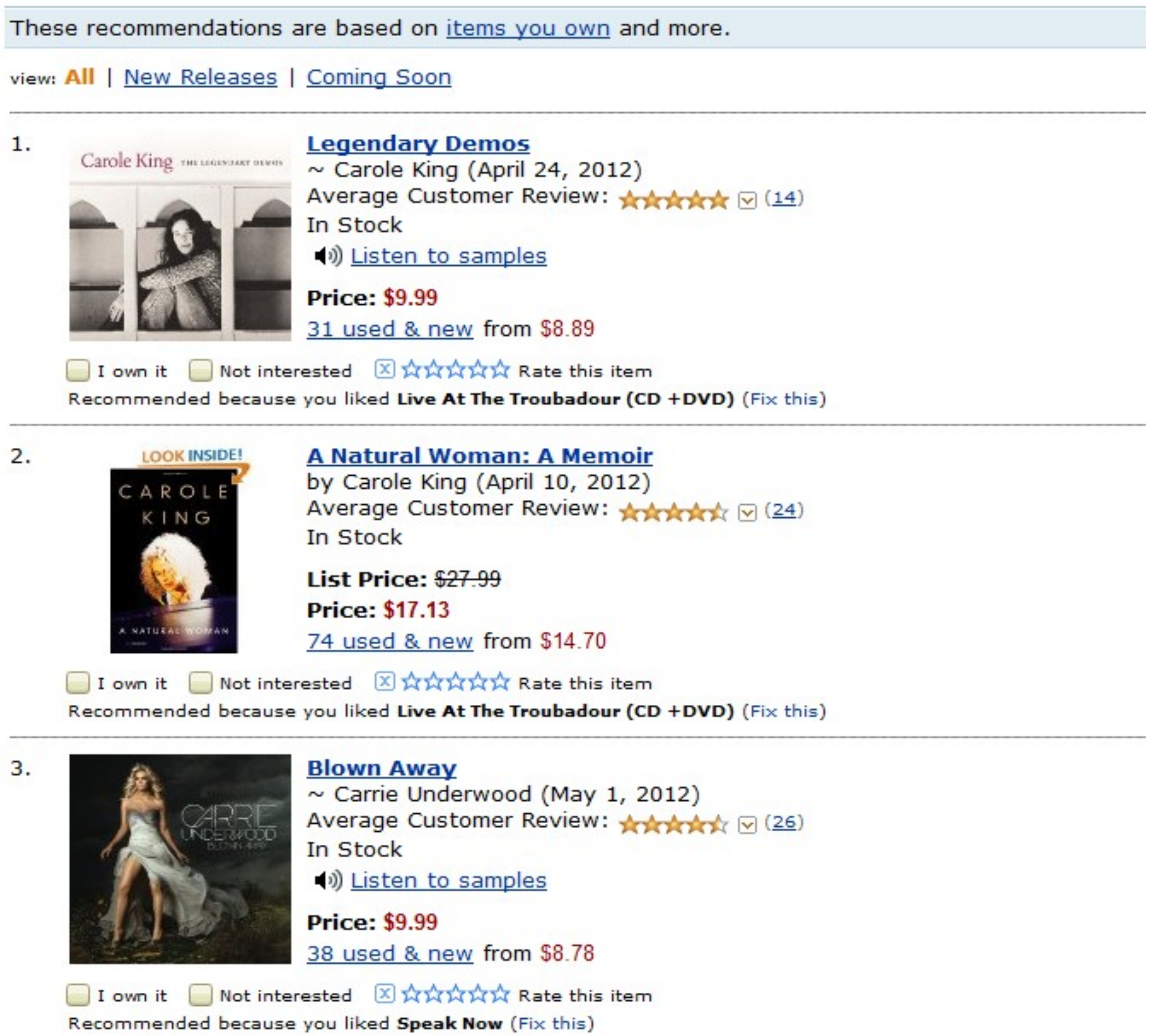
- Introduction and Background
- POI Recommendation
- Successive POI Recommendation
- Gradient Boosting Factorization Machines
- Conclusion

Outline

- Introduction and Background
- POI Recommendation
- Successive POI Recommendation
- Gradient Boosting Factorization Machines
- Conclusion

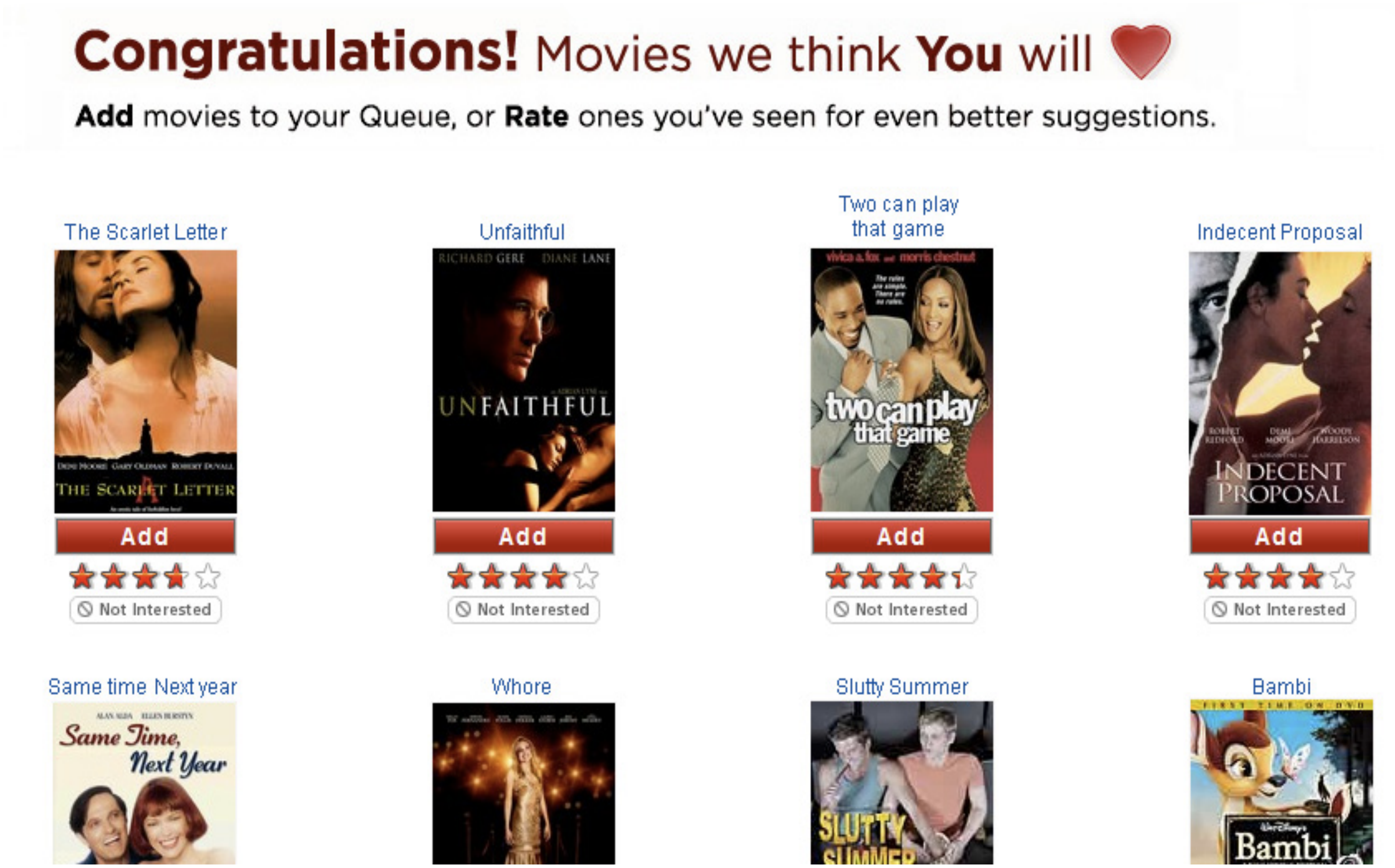
Recommender Systems

Recommender Systems



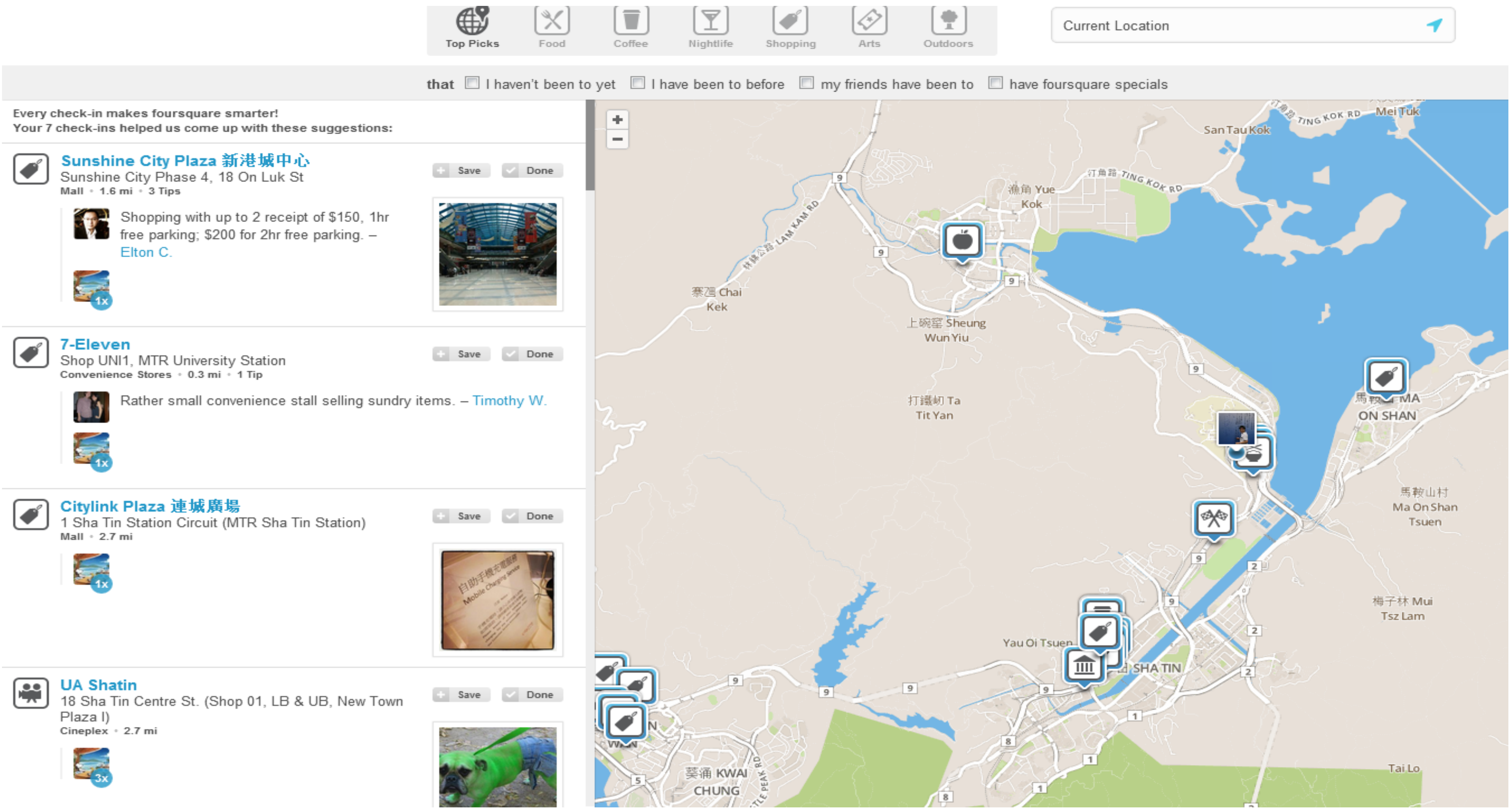
Recommender Systems

Recommender Systems



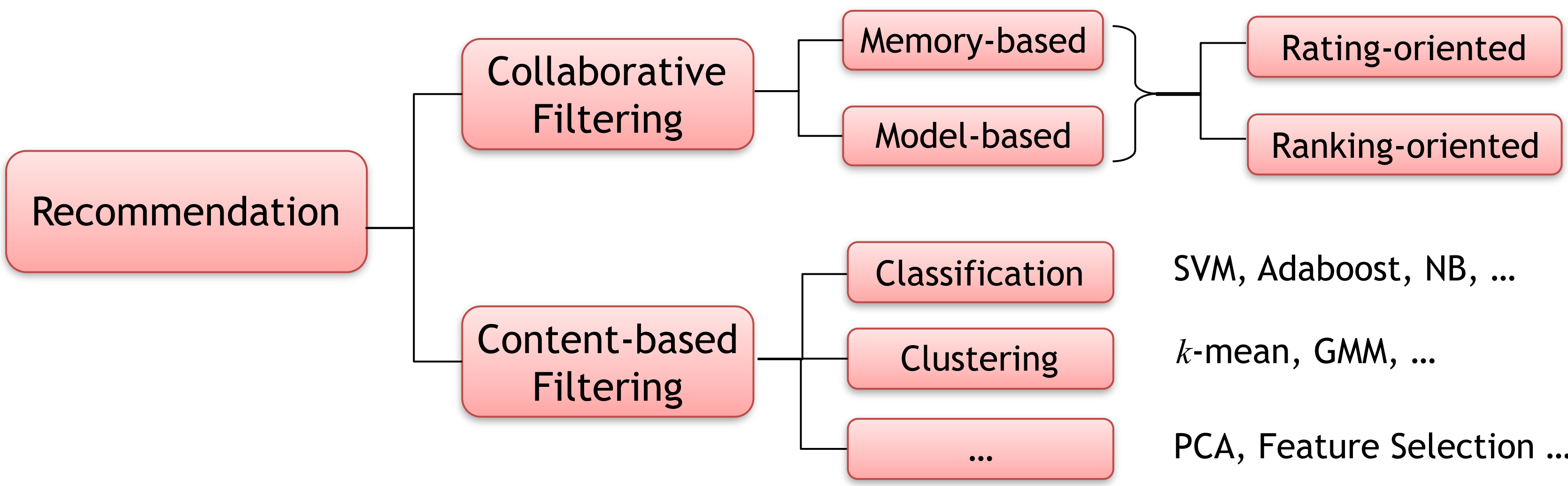
Recommender Systems

Recommender Systems



Recommender Systems

Recommendation Approaches



- Collaborative filtering
  - Use user-item rating matrix to predict rating/ranking
  - Simple in data collection
- Content-based filtering
  - Users' preference expressed in intrinsic features
  - Difficult in feature representation

Memory-based Collaborative Filtering

Memory-based Collaborative Filtering

- Leverage similar users'/items' ratings

	v_1	v_2	v_3	v_4	v_5	v_6
u_1		5	2		3	
u_2	4			3		4
u_3			2			2
u_4	5			3		
u_5		5	5			3

Memory-based Collaborative Filtering

- Leverage similar users'/items' ratings
- Pros
  - Simple to implement
  - Clear interpretation

	v_1	v_2	v_3	v_4	v_5	v_6
u_1		5	2		3	
u_2	4			3		4
u_3			2			2
u_4	5			3		
u_5		5	5			3

Memory-based Collaborative Filtering

- Leverage similar users'/items' ratings
- Pros
  - Simple to implement
  - Clear interpretation
- Cons
  - High computational cost
  - Prone to sparseness problem

	v_1	v_2	v_3	v_4	v_5	v_6
u_1		5	2		3	
u_2	4			3		4
u_3			2			2
u_4	5			3		
u_5		5	5			3

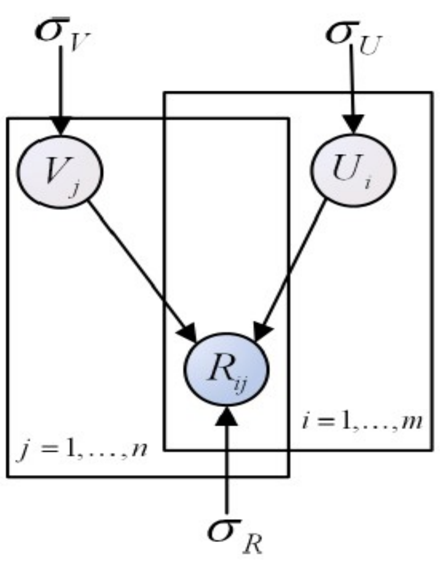
Model-based Collaborative Filtering

- Train a pre-defined model
- Efficient in prediction time
- Usually outperform memory-based methods
- Successful methods:
    - Probabilistic Matrix Factorization (PMF) [Salakhutdinov et al., 2007]
    - Bayesian Personalized Ranking (BPR) [Rendle et al, 2009]

PMF

- Use two low rank matrices U and V to approximate the rating matrix R:

$$R \approx U^T V, U \in \mathbb{R}^{k\times m}, V \in \mathbb{R}^{k\times n}$$



- Conditional distribution over observed ratings: $p(R|U,V,\sigma_R^2) = \prod_{i=1}^{m}\prod_{j=1}^{n}[\mathcal{N}(R_{ij}|U_iV_j^T,\sigma_R^2)]^{I_{ij}^R}$

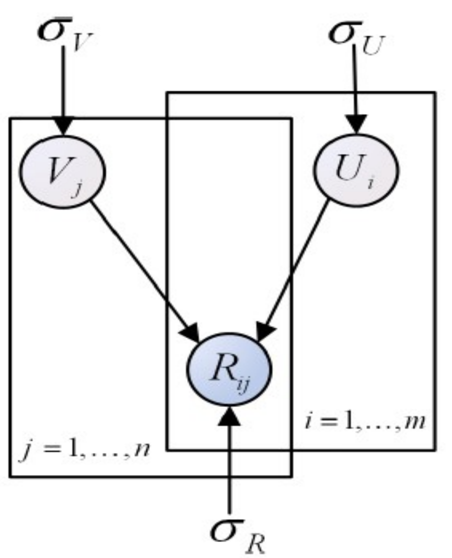
- Zero-mean spherical Gaussian priors on user and item feature vectors:

$$p(U|\sigma_U^2) = \prod_{i=1}^{m}\mathcal{N}(U_i|0,\sigma_U^2\mathbf{I}), p(V|\sigma_V^2) = \prod_{j=1}^{n}\mathcal{N}(V_j|0,\sigma_V^2\mathbf{I}).$$

PMF

- Maximize the posterior:

$$p(U,V|R,\sigma_R^2,\sigma_U^2,\sigma_V^2) \propto p(R|U,V,\sigma_R^2)p(U|\sigma_U^2)p(V|\sigma_V^2)$$



- The objective function is:

$$\min_{U,V} \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^R (R_{ij} - U_i V_j^T)^2 + \frac{\lambda_1}{2}\|U\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2$$

BPR

- A ranking-oriented method
- Construct the pairwise training set
  $$D_S = \{(u, i, j) | u \in U \wedge i \in I_u^+ \wedge j \in I \setminus I_u^+\}$$
  - a user u prefers i (observed) over j (unobserved)
- Maximize the posterior:
  $$\prod_{(u,i,j) \in D_S} p(i >_u j | \Theta) p(\Theta)$$

BPR

- Define the prob. a user prefers i over j as:

$$p(i >_u j|\Theta) = \sigma(\hat{x}_{ui} - \hat{x}_{uj})$$

$$\hat{x}_{ui} = U_u^T V_i$$

- Finally we maximize:

$$\sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{ui} - \hat{x}_{uj}) - \lambda_\Theta \|\Theta\|_F^2$$

Problems in Traditional Recommendation Methods

Problems in Traditional Recommendation Methods

- Data Sparsity
    - Extreme sparse in some applications such as POI recommendation
    - How to alleviate data sparsity problem

Problems in Traditional Recommendation Methods

- Data Sparsity
  - Extreme sparse in some applications such as POI recommendation
  - How to alleviate data sparsity problem
- Context information
  - Abundant context information available: age, category, special date, etc.
  - How to employ context information

Outline

- Introduction and Background
- POI Recommendation
- Successive POI Recommendation
- Gradient Boosting Factorization Machines
- Conclusion

Location-based Social Networks (LBSNs)

Growth of Location-based Services (LBS)

- Almost one fifth of the world's six billion mobile users are already using LBS

- 26% users use the technology to find restaurants and entertainment venues

- 74% of smartphone owners use LBS.

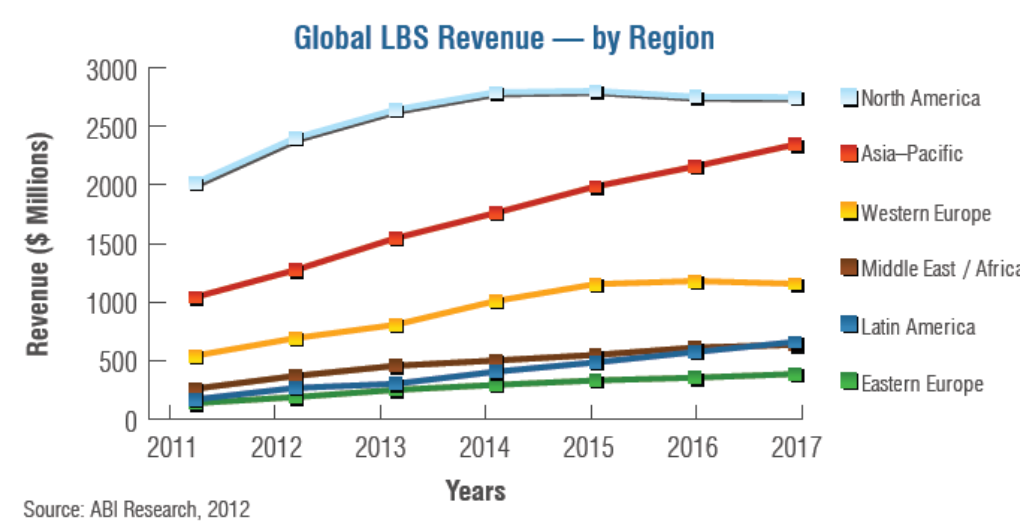
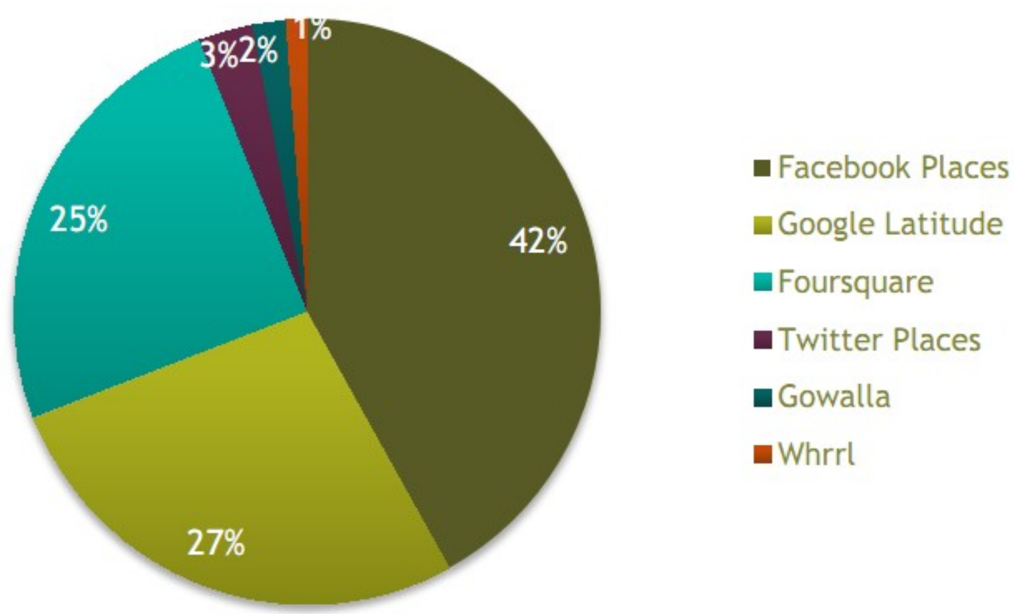


Figure 2. Projected LBS services revenue by region (2011-2017)[6]

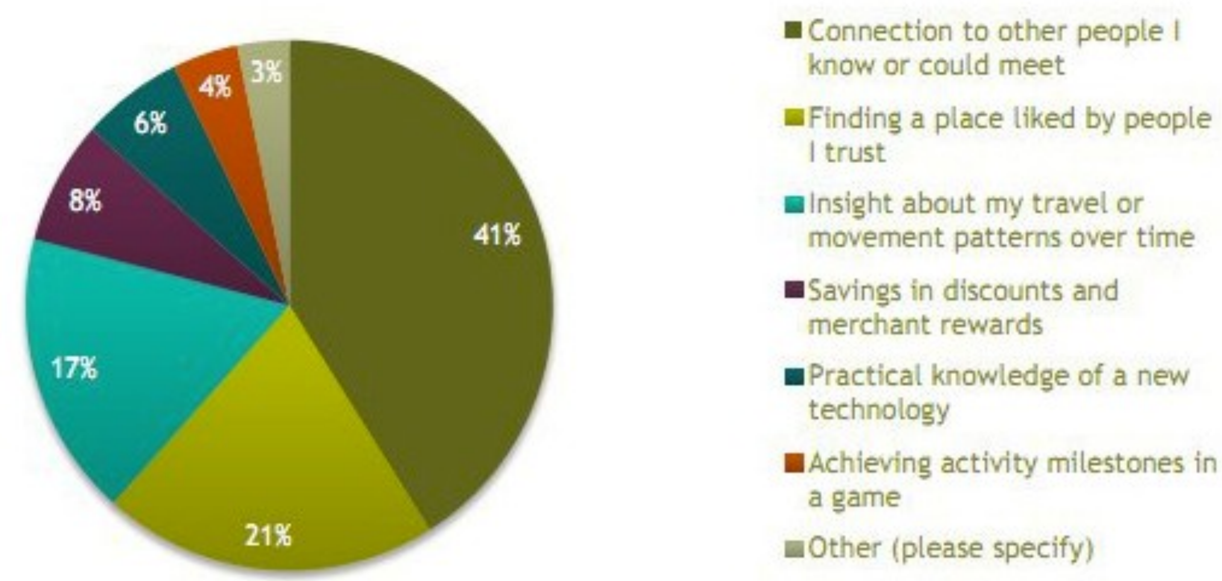
Check-in Becomes a Life-style

Check-in Becomes a Life-style

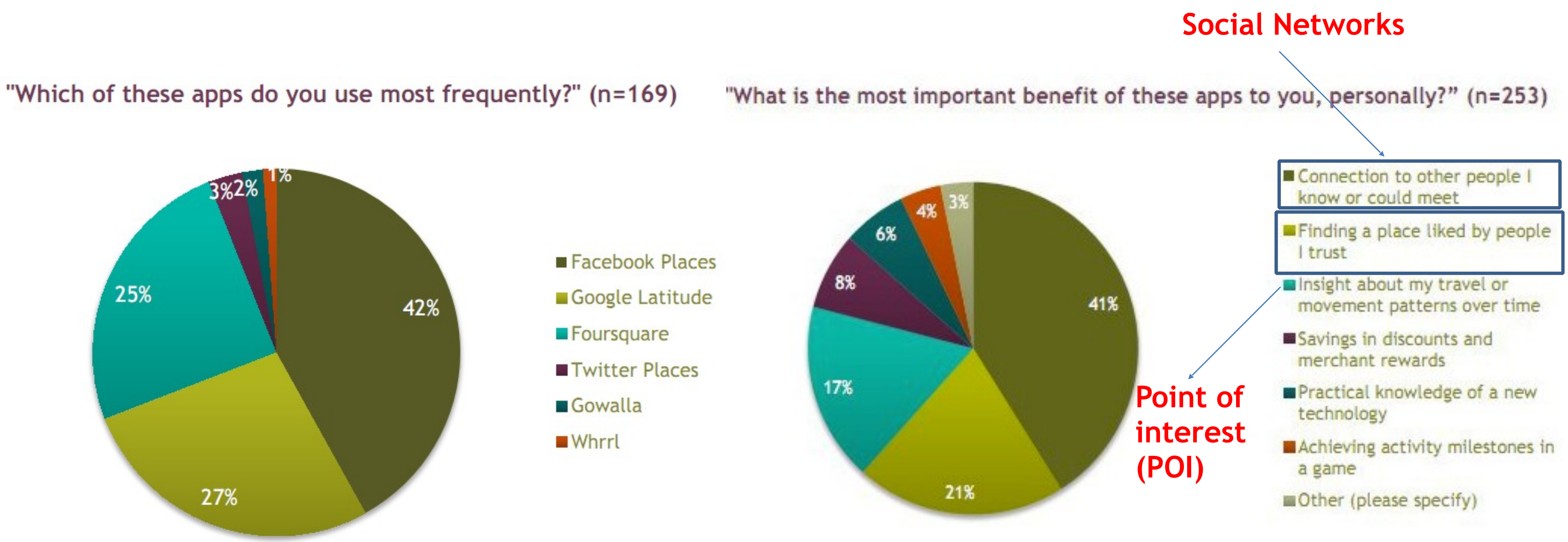
"Which of these apps do you use most frequently?" (n=169)



"What is the most important benefit of these apps to you, personally?" (n=253)



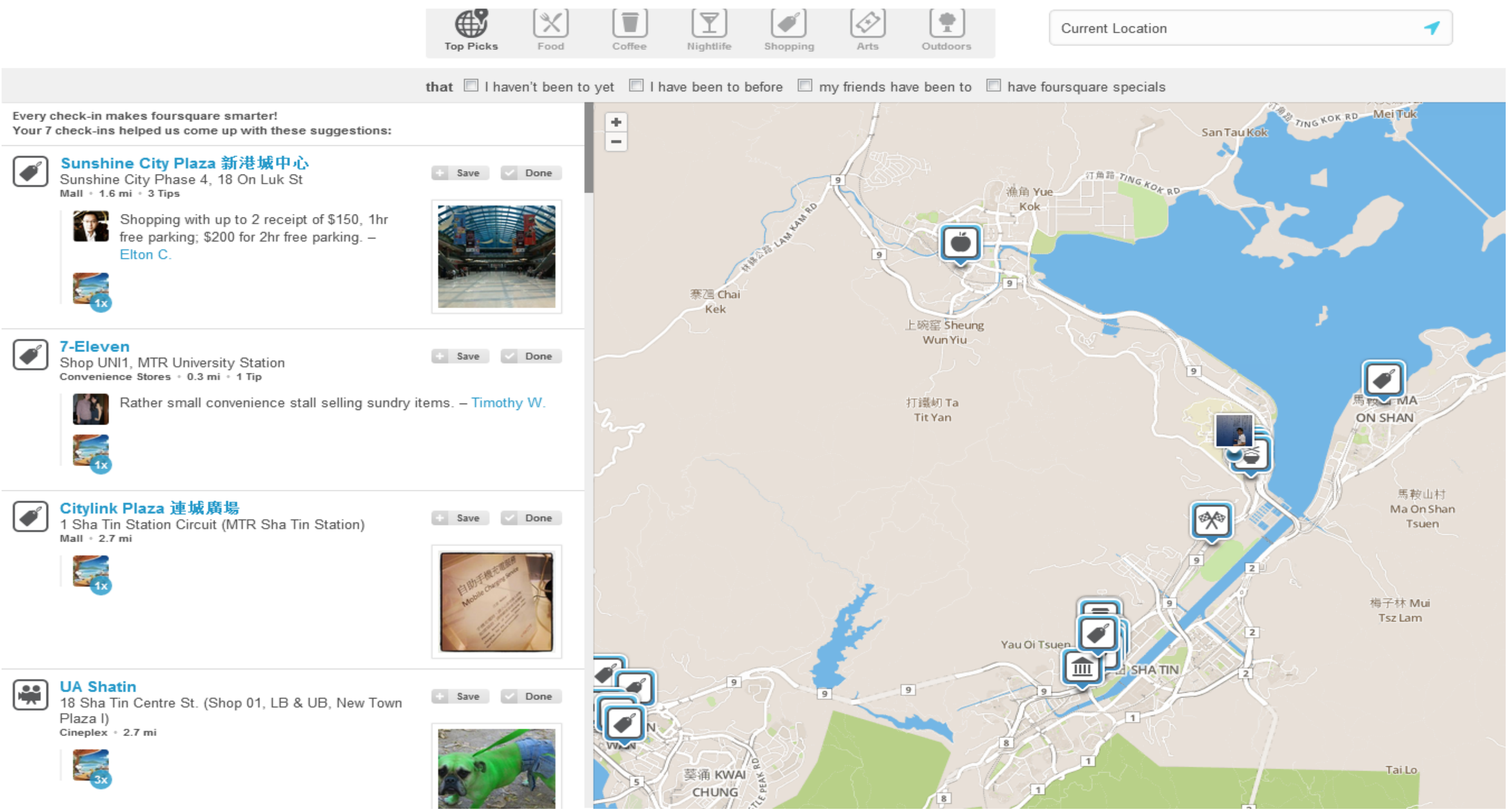
Check-in Becomes a Life-style



Our Focus: POI Recommendation

Our Focus: POI Recommendation

- Help users explore their surroundings

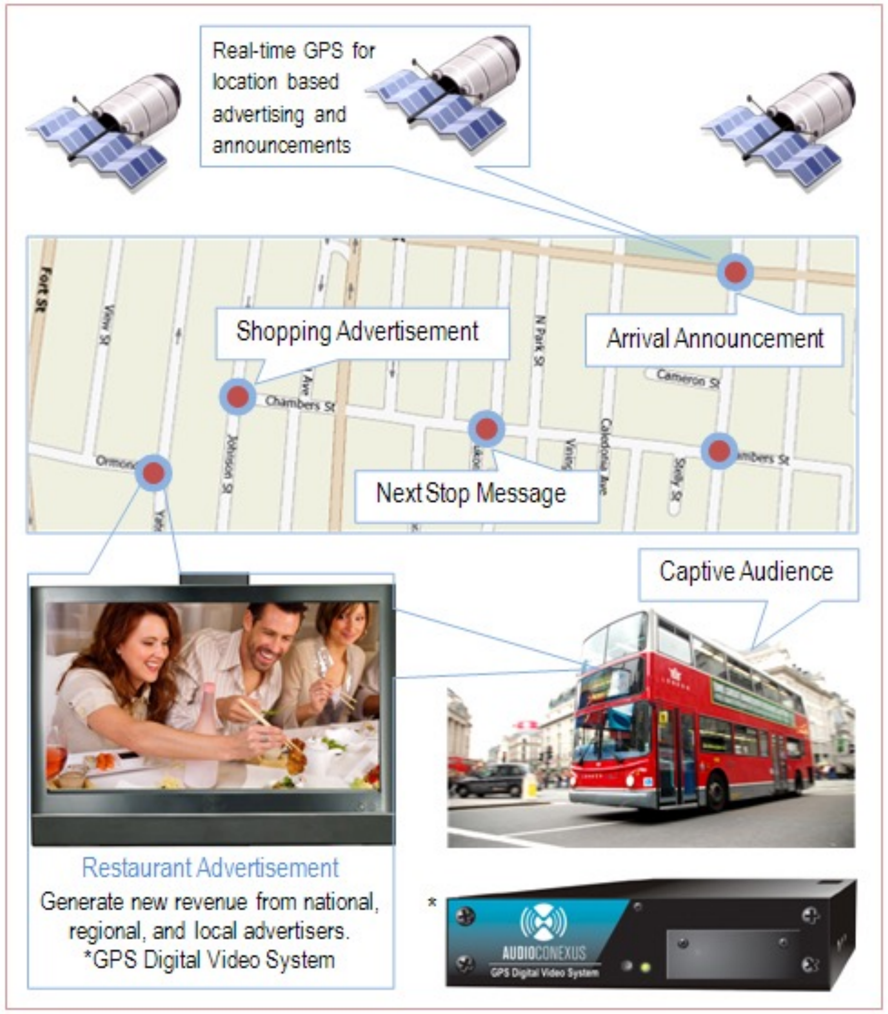


Our Focus: POI Recommendation

- Help users explore their surroundings

Our Focus: POI Recommendation

- Help users explore their surroundings
- Help 3rd-party developers provide personalized services
  - Advertisements
  - Coupons
  - Traffic statistics



Challenges

- Large dataset
  - 4,128,714 check-ins from 53,944 users on 367,149 locations for Gowalla
- Sparsity : density of our dataset is only 0.0208%
  - Matrix Factorization can be inaccurate

	l_1	l_2	l_3	l_4	l_5	l_6	$\cdots$	$l_{\vert\mathcal{L}\vert-1}$	$l_{\vert\mathcal{L}\vert}$
u_1	?	?	164	?	1	?	$\cdots$	?	1
u_2	40	2	?	?	?	1	$\cdots$	?	?
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ldots$	$\vdots$	$\vdots$
$u_{\vert\mathcal{U}\vert-1}$	?	?	1	1	?	?	$\cdots$	2	?
$u_{\vert\mathcal{U}\vert}$	?	2	?	?	1	?	$\cdots$	?	10

Figure 1: User-location check-in frequency matrix.

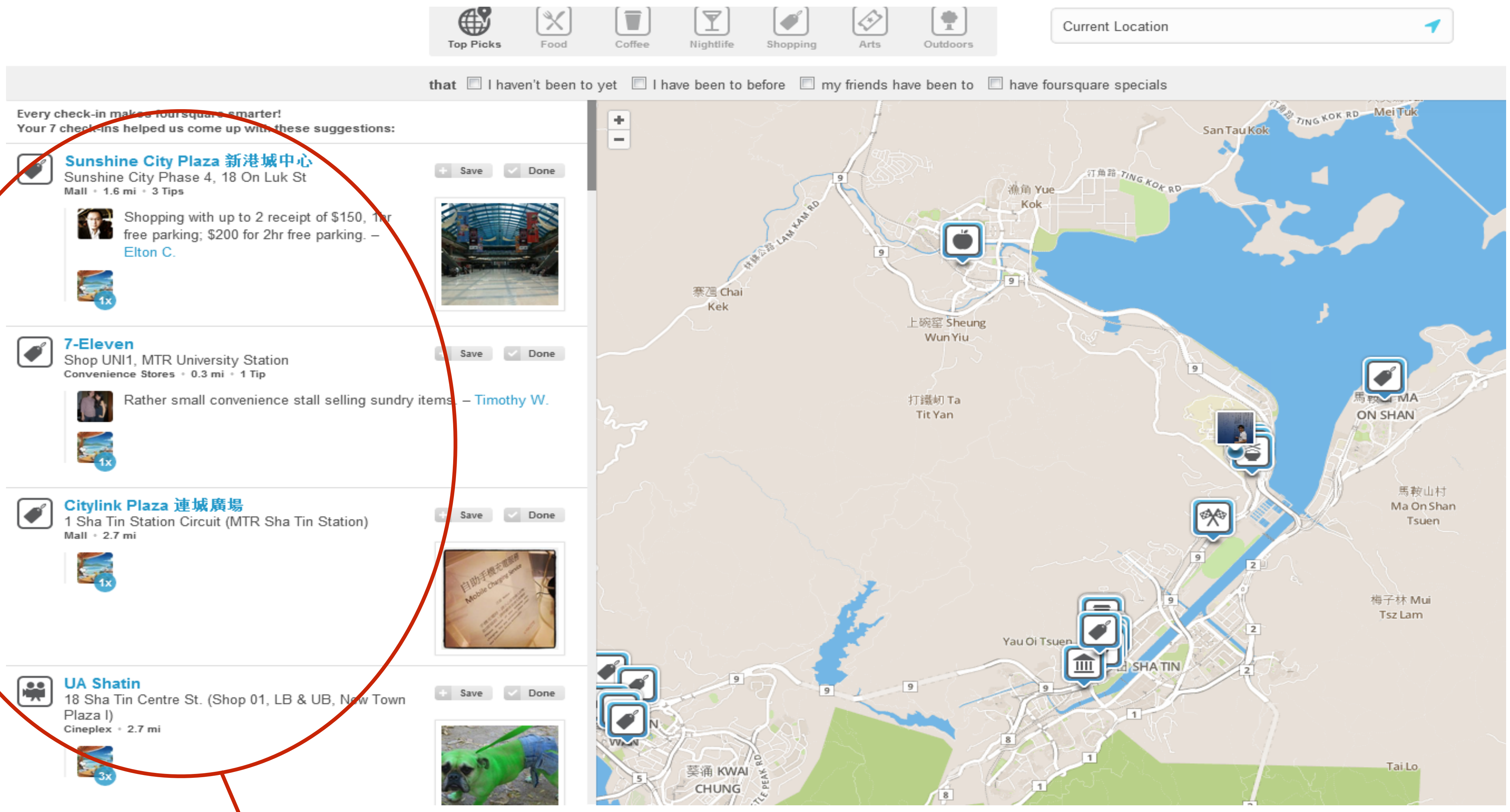
Geographical Influence

Geographical Influence

Er… a little
far..

Top-k Ranking

Top-k Ranking

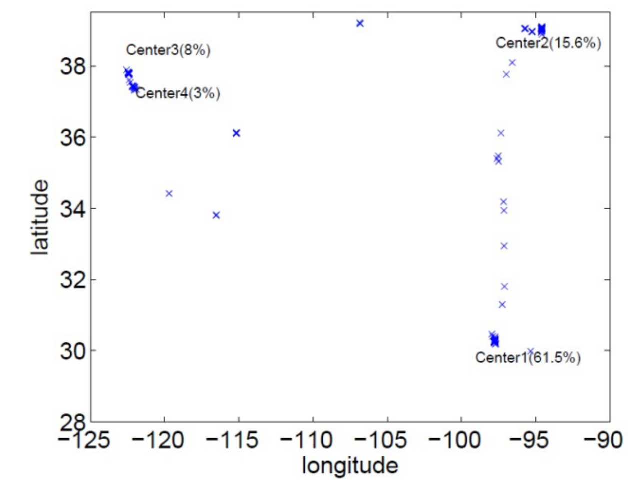


users care more about top results

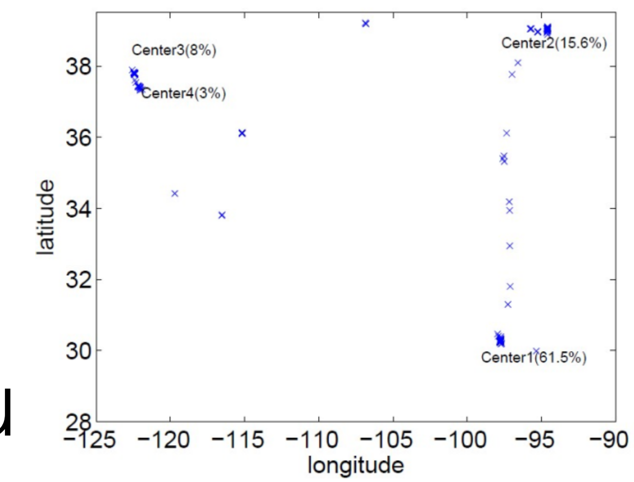
Our Proposal

- Multi-center Gaussian Model (MGM) to capture the geographical influence
- Fused matrix factorization framework with MGM
- Propose two methods based on BPR to address geographical influence and top-k ranking

Multi-center Gaussian Model

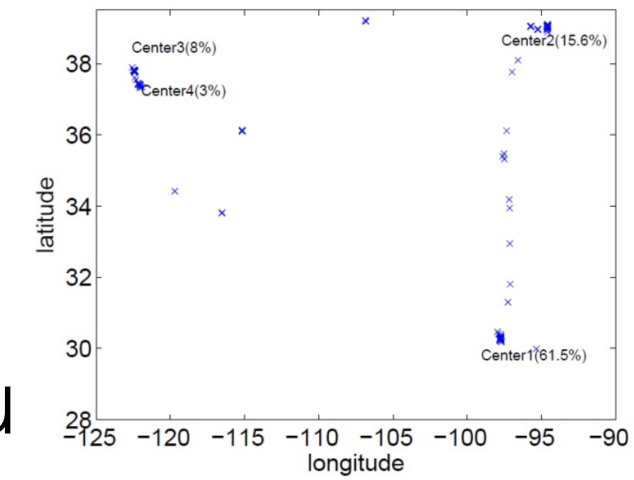


Multi-center Gaussian Model



- Notation
  - C_u: multi-center set for user u
  - f_{c_u}: total frequency at center c_u for user u
  - $\mathcal{N}(l|\mu_{c_u}, \Sigma_{c_u})$ is : the pdf of Gaussian distribution, μ_{c_u} and Σ_{c_u} denote the mean and covariance matrices of regions around center c_u

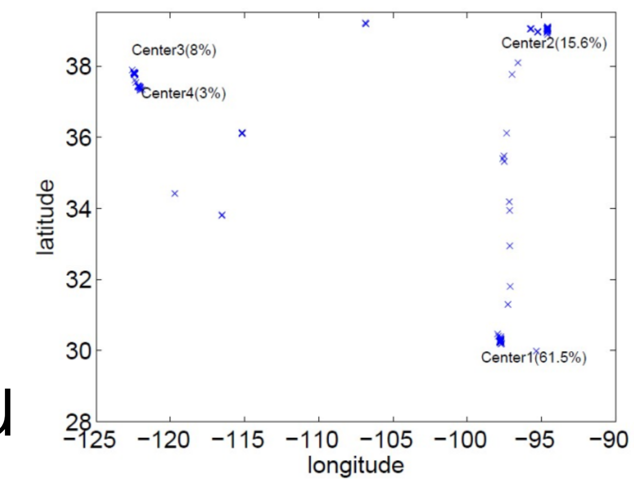
Multi-center Gaussian Model



- Notation
  - C_u: multi-center set for user u
  - f_{c_u}: total frequency at center c_u for user u
  - $\mathcal{N}(l|\mu_{c_u}, \Sigma_{c_u})$ is : the pdf of Gaussian distribution, μ_{c_u} and Σ_{c_u} denote the mean and covariance matrices of regions around center c_u
- The probability a user u visiting a location l given c_u defined as:

$$P(l|C_u) = \sum_{c_u=1}^{|C_u|} P(l \in c_u) \frac{f_{c_u}^{\alpha}}{\sum_{i \in C_u} f_i^{\alpha}} \frac{\mathcal{N}(l|\mu_{c_u}, \Sigma_{c_u})}{\sum_{i \in C_u} \mathcal{N}(l|\mu_i, \Sigma_i)}$$

Multi-center Gaussian Model

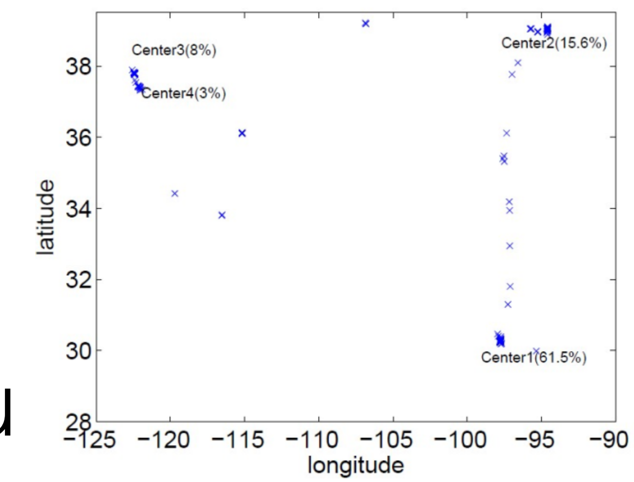


- Notation
  - C_u: multi-center set for user u
  - f_{c_u}: total frequency at center c_u for user u
  - $\mathcal{N}(l|\mu_{c_u}, \Sigma_{c_u})$ is : the pdf of Gaussian distribution, μ_{c_u} and Σ_{c_u} denote the mean and covariance matrices of regions around center c_u
- The probability a user u visiting a location l given c_u defined as:

$$P(l|C_u) = \sum_{c_u=1}^{|C_u|} P(l \in c_u) \frac{f_{c_u}^{\alpha}}{\sum_{i \in C_u} f_i^{\alpha}} \frac{\mathcal{N}(l|\mu_{c_u}, \Sigma_{c_u})}{\sum_{i \in C_u} \mathcal{N}(l|\mu_i, \Sigma_i)}$$

$\propto 1/dist(l, c_u)$

Multi-center Gaussian Model



- Notation
  - C_u: multi-center set for user u
  - f_{c_u}: total frequency at center c_u for user u
  - $\mathcal{N}(l|\mu_{c_u}, \Sigma_{c_u})$ is : the pdf of Gaussian distribution, μ_{c_u} and Σ_{c_u} denote the mean and covariance matrices of regions around center c_u
- The probability a user u visiting a location l given c_u defined as:

$$P(l|C_u) = \sum_{c_u=1}^{|C_u|} P(l \in c_u) \frac{f_{c_u}^{\alpha}}{\sum_{i \in C_u} f_i^{\alpha}} \frac{\mathcal{N}(l|\mu_{c_u}, \Sigma_{c_u})}{\sum_{i \in C_u} \mathcal{N}(l|\mu_i, \Sigma_i)}$$

$\propto 1/dist(l, c_u)$

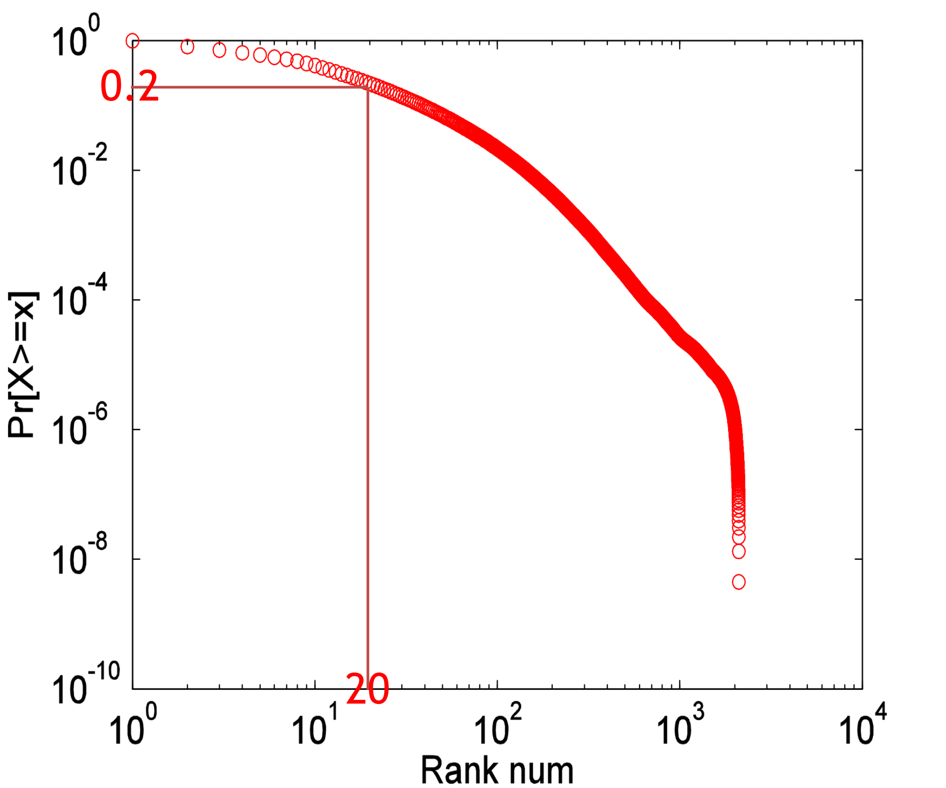
norm effect of check in freq on center c_u

Multi-center Discovering Algorithm

A greedy clustering algorithm is proposed due to Pareto principle (top 20 locations cover about 80% check-ins)

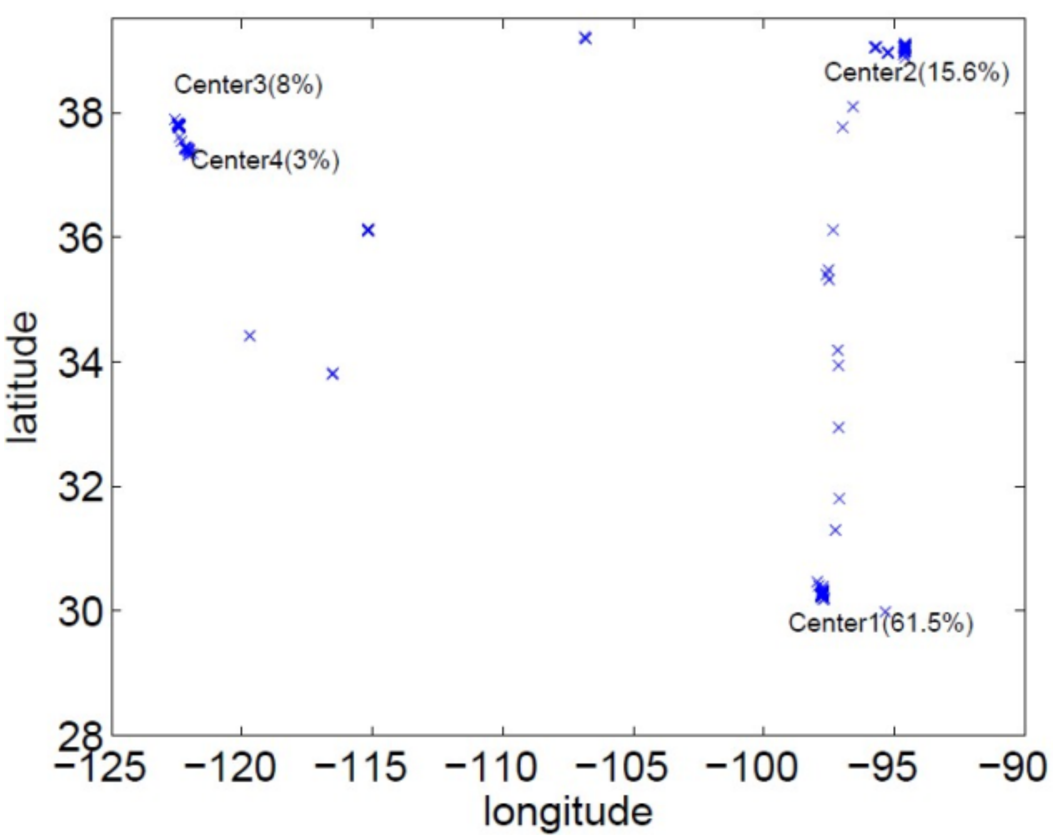
Multi-center Discovering Algorithm

A greedy clustering algorithm is proposed due to Pareto principle (top 20 locations cover about 80% check-ins)



Multi-center Discovering Algorithm

A greedy clustering algorithm is proposed due to Pareto principle (top 20 locations cover about 80% check-ins)

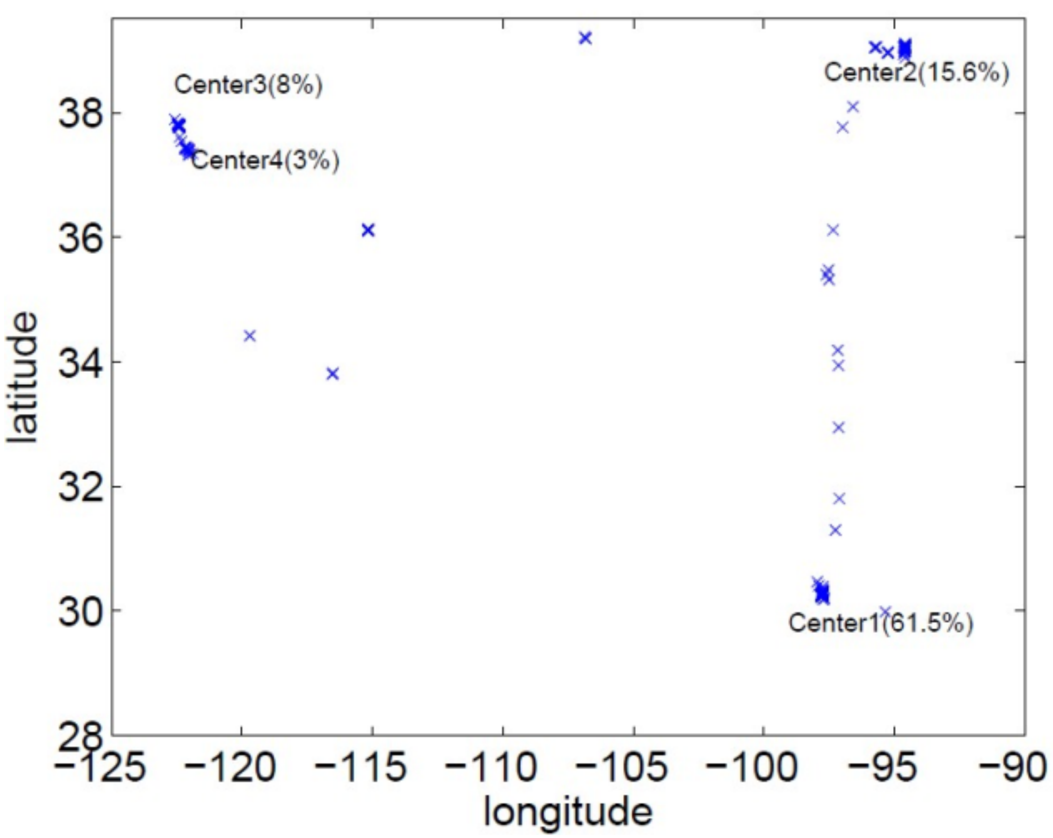


Algorithm 1 Multi-center Discovering Algorithm

for all user i in the user set $\mathcal{U}$ **do**
  Rank all check-in locations in $|\mathcal{L}|$ according to visiting frequency
  $\forall l_k \in L$, set $l_k.center = -1$;
  Center_list = $\emptyset$; center_no = 0;
  for $i = 1 \rightarrow |L|$ **do**
    if $l_i.center == -1$ **then**
      center_no++; Center = $\emptyset$; Center.total_freq = 0;
      Center.add(l_i); Center.total_freq += l_i.freq;
      for $j = i + 1 \rightarrow |L|$ **do**
        if $l_j.center == -1$ *and* $dist(l_i, l_j) \leq d$ **then**
          $l_j.center$ = center_no; Center.add(l_j);
          Center.total_freq += l_j.freq;
        end if
      end for
      if Center.total_freq $\geq |u_i|$.total_freq * θ **then**
        Center_list.add(Center);
      end if
    end if
  end for
  RETURN Center_list for user i;
end for

Multi-center Discovering Algorithm

A greedy clustering algorithm is proposed due to Pareto principle (top 20 locations cover about 80% check-ins)



Algorithm 1 Multi-center Discovering Algorithm

---

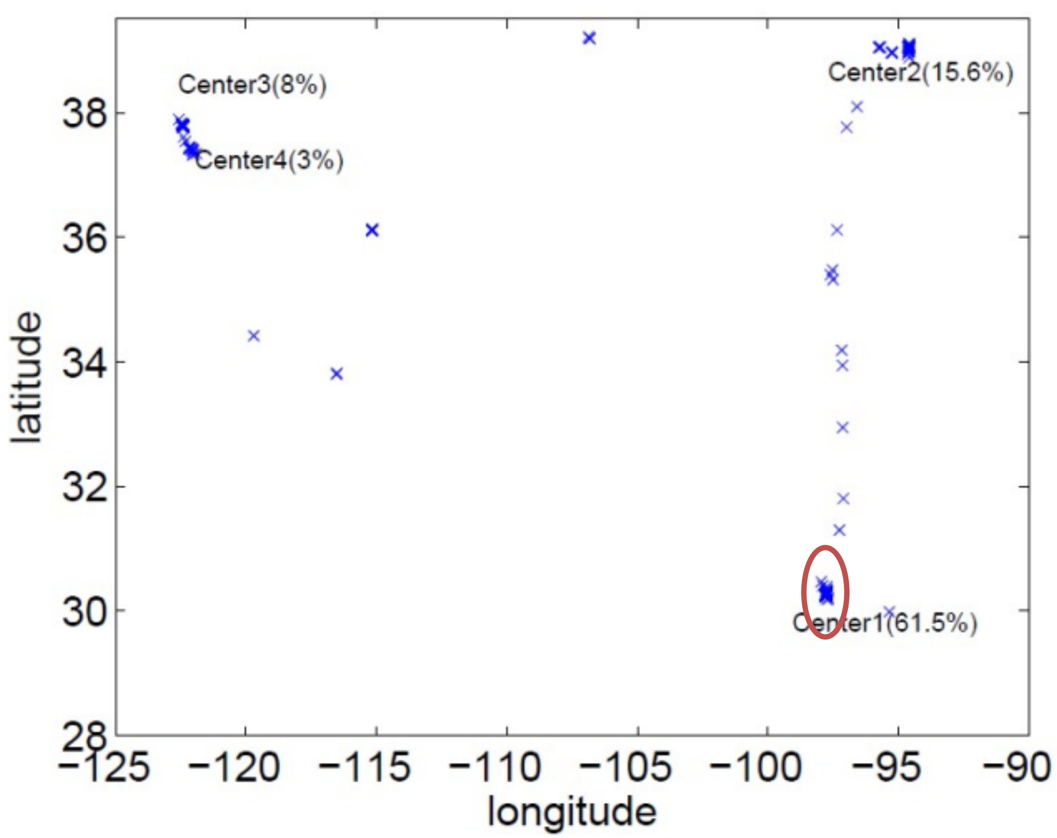
for all user i in the user set $\mathcal{U}$ **do**
    Rank all check-in locations in $|\mathcal{L}|$ according to visiting frequency
    $\forall l_k \in L$, set $l_k.center = -1$;
    Center_list = $\emptyset$; center_no = 0;
    for $i = 1 \rightarrow |L|$ **do**
        if $l_i.center == -1$ **then**
            center_no++; Center = $\emptyset$; Center.total_freq = 0;
            Center.add(l_i); Center.total_freq += l_i.freq;
            for $j = i + 1 \rightarrow |L|$ **do**
                if $l_j.center == -1$ *and* $dist(l_i, l_j) \leq d$ **then**
                    $l_j.center$ = center_no; Center.add(l_j);
                    Center.total_freq += l_j.freq;
                end if
            end for
            if Center.total_freq $\geq |u_i|$.total_freq * θ **then**
                Center_list.add(Center);
            end if
        end if
    end for
    RETURN Center_list for user i;
end for

---

search centers

Multi-center Discovering Algorithm

A greedy clustering algorithm is proposed due to Pareto principle (top 20 locations cover about 80% check-ins)



Algorithm 1 Multi-center Discovering Algorithm

for all user i in the user set $\mathcal{U}$ **do**
  Rank all check-in locations in $|\mathcal{L}|$ according to visiting frequency
  $\forall l_k \in L$, set $l_k.center = -1$;
  Center_list = $\emptyset$; center_no = 0;
  for $i = 1 \to |L|$ **do**
    if $l_i.center == -1$ **then**
      center_no++; Center = $\emptyset$; Center.total_freq = 0;
      Center.add(l_i); Center.total_freq += l_i.freq;
      for $j = i + 1 \to |L|$ **do**
        if $l_j.center == -1$ *and* $dist(l_i, l_j) \le d$ **then**
          $l_j.center$ = center_no; Center.add(l_j);
          Center.total_freq += l_j.freq;
        end if
      end for
      if Center.total_freq $\ge |u_i|$.total_freq * θ **then**
        Center_list.add(Center);
      end if
    end if
  end for
  RETURN Center_list for user i;
end for

search centers

Fused Matrix Factorization with MGM (FMFMGM)

$$P_{ul} = P(F_{ul}) \cdot P(l|C_u)$$

Fused Matrix Factorization with MGM (FMFMGM)

- Traditional Matrix Factorization (MF) only model users' preference on locations

User

	l_1	l_2	l_3	l_4	l_5	l_6	$\cdots$	$l_{\lvert\mathcal{L}\rvert-1}$	$l_{\lvert\mathcal{L}\rvert}$
u_1	?	?	164	?	1	?	$\cdots$	?	1
u_2	40	2	?	?	?	1	$\cdots$	?	?
$u_{\lvert\mathcal{U}\rvert-1}$	?	?	1	1	?	?	$\cdots$	2	?
$u_{\lvert\mathcal{U}\rvert}$	?	2	?	?	1	?	$\cdots$	?	10

$$F = UL^T$$

Location

$$P_{ul} = P(F_{ul}) \cdot P(l|C_u)$$

encode user preference
based on MF

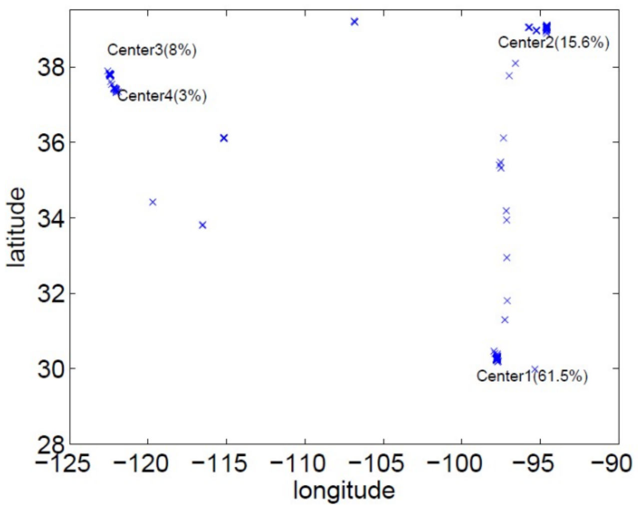
Fused Matrix Factorization with MGM (FMFMGM)

- Traditional Matrix Factorization (MF) only model users' preference on locations
- MGM only models geographical influence

$$P_{ul} = P(F_{ul}) \cdot P(l|C_u)$$

encode user preference based on MF

calculated by MGM



Fused Matrix Factorization with MGM (FMFMGM)

- Traditional Matrix Factorization (MF) only model users' preference on locations
- MGM only models geographical influence
- We can fuse both of them

$$P_{ul} = P(F_{ul}) \cdot P(l|C_u)$$

prob. user u visit location l

encode user preference based on MF

calculated by MGM

BPR Location Recommendation

- BPRLR1: same as the previous fusion method

$$P_{ul} = P(F_{ul}) \cdot P(l|C_u)$$

BPR Location Recommendation

- BPRLR1: same as the previous fusion method

$$P_{ul} = P(F_{ul}) \cdot P(l|C_u)$$

encode user preference
based on BPR

BPR Location Recommendation

BPR Location Recommendation

- BPRLR2: reconstruct the training pairwise location set
  - Maximize the difference between visited location and unvisited nearby location

BPR Location Recommendation

- BPRLR2: reconstruct the training pairwise location set
  - Maximize the difference between visited location and unvisited nearby location

$$S' = \{(u, i, j) | u \in \mathcal{U}, i \in \mathcal{L}_u^+ \wedge j \in N_u \setminus \mathcal{L}_u^+\}$$

$$N_u = \{l | P(l|C_u) > 0\}$$

BPR Location Recommendation

- BPRLR2: reconstruct the training pairwise location set
    - Maximize the difference between visited location and unvisited nearby location

$$S' = \{(u,i,j)|u \in \mathcal{U}, i \in \mathcal{L}_u^+ \wedge j \in N_u \setminus \mathcal{L}_u^+\}$$

$$N_u = \{l|P(l|C_u) > 0\}$$

calculated by MGM

Dataset

- Two publicly available data sets: Foursquare and Gowalla

Table 3.1: Basic statistics of the Gowalla and Foursquare dataset for POI recommendation

$\#U$	$\#L$	$\#E$
$53,944$	$367,149$	$306,958$
$\#\widetilde{U}$	$\#\widetilde{L}$	$\#\widetilde{E}$
51.33	7.54	11.38
$\#\text{max. } U$	$\#\text{max. } L$	$\#\text{max. } E$
$2,145$	$3,581$	$2,366$

(a) Gowalla

$\#U$	$\#L$
$6,084$	$37,976$
$\#\widetilde{U}$	$\#\widetilde{L}$
35.98	5.76
$\#\text{max. } U$	$\#\text{max. } L$
182	985

(b) Foursquare

Results

Results

Table III. Performance Comparisons on the Gowalla dataset with $K = 30$

Ratio	Metrics	Dimension = 30							
		MGM	PMF	PMFSR	PFM	FMFMGM	BPR	BPRLR1	BPRLR2
70%	P@5	0.0317	0.0148	0.0158	0.0173	0.0672	0.0674	**0.0802**	0.0517
	Improve	153.00%	441.89%	407.59%	363.58%	19.35%	18.99%		55.13%
	R@5	0.0113	0.0033	0.0035	0.0040	0.0212	0.0199	**0.0270**	0.0175
	Improve	138.94%	718.18%	671.43%	575.00%	27.36%	35.68%		54.29%
	P@10	0.0273	0.0162	0.0174	0.0173	0.0656	0.0643	**0.0700**	0.0628
	Improve	156.41%	332.10%	302.30%	304.62%	6.71%	8.86%		11.46%
	R@10	0.0194	0.0075	0.0080	0.0084	0.0408	0.0382	**0.0465**	0.0408
	Improve	260.82%	833.33%	775.00%	733.33%	71.57%	83.25%		71.57%
80%	P@5	0.0263	0.0106	0.011	0.0114	0.0486	0.0488	**0.0551**	0.0348
	Improve	109.51%	419.81%	400.91%	383.33%	13.37%	12.91%		58.33%
	R@5	0.0141	0.0035	0.0037	0.0039	0.0218	0.0210	**0.0263**	0.0172
	Improve	86.52%	651.43%	610.81%	574.36%	20.64%	25.24%		52.91%
	P@10	0.0226	0.0115	0.0117	0.0117	0.0472	0.0450	**0.0479**	0.0432
	Improve	111.95%	316.52%	309.40%	309.40%	1.48%	6.44%		10.88%
	R@10	0.0244	0.0079	0.0081	0.0085	0.0424	0.0386	**0.0456**	0.0407
	Improve	86.89%	477.22%	462.96%	436.47%	7.55%	18.13%		12.04%

Table V. Performance Comparisons on the Foursquare dataset with $K = 30$

Ratio	Metrics	Dimension = 30						
		MGM	PMF	PFM	FMFMGM	BPR	BPRLR1	BPRLR2
70%	P@5	0.0409	0.0621	0.0718	0.1201	0.1086	0.1484	**0.1783**
	Improve	335.94%	187.12%	148.33%	48.46%	64.18%	20.15%	
	R@5	0.0306	0.0277	0.0312	0.0594	0.0528	0.0763	**0.0901**
	Improve	194.44%	225.27%	188.78%	51.68%	70.64%	18.09%	
	P@10	0.0373	0.0638	0.0663	0.1166	0.1107	0.1522	**0.1698**
	Improve	355.23%	166.14%	156.11%	45.63%	53.39%	11.56%	
	R@10	0.0531	0.0574	0.0622	0.1166	0.1070	0.1568	**0.1728**
	Improve	225.42%	201.05%	177.81%	48.20%	61.50%	10.20%	
80%	P@5	0.0288	0.0450	0.0482	0.0833	0.0820	0.1050	**0.1287**
	Improve	346.88%	186.00%	167.01%	54.50%	56.95%	22.57%	
	R@5	0.0332	0.0306	0.0364	0.0640	0.0606	0.0834	**0.0998**
	Improve	200.60%	226.14%	174.18%	55.94%	64.69%	19.66%	
	P@10	0.0265	0.0478	0.0512	0.0811	0.0796	0.1053	**0.1227**
	Improve	363.02%	156.69%	139.65%	51.29%	54.15%	16.52%	
	R@10	0.0586	0.0657	0.0677	0.1242	0.1176	0.1658	**0.1898**
	Improve	223.89%	188.89%	180.35%	52.82%	61.39%	14.48%	

Results

Table III. Performance Comparisons on the Gowalla dataset with $K = 30$

Ratio	Metrics	Dimension = 30							
		MGM	PMF	PMFSR	PFM	PMFMGM	BPR	BPRLR1	BPRLR2
70%	P@5	0.0317	0.0148	0.0158	0.0173	0.0672	0.0674	**0.0802**	0.0517
	Improve	153.00%	441.89%	407.59%	363.58%	19.35%	18.99%		55.13%
	R@5	0.0113	0.0033	0.0035	0.0040	0.0212	0.0199	**0.0270**	0.0175
	Improve	138.94%	718.18%	671.43%	575.00%	27.36%	35.68%		54.29%
	P@10	0.0273	0.0162	0.0174	0.0173	0.0656	0.0643	**0.0700**	0.0628
	Improve	156.41%	332.10%	302.30%	304.62%	6.71%	8.86%		11.46%
	R@10	0.0194	0.0075	0.0080	0.0084	0.0408	0.0382	**0.0465**	0.0408
	Improve	260.82%	833.33%	775.00%	733.33%	71.57%	83.25%		71.57%
80%	P@5	0.0263	0.0106	0.011	0.0114	0.0486	0.0488	**0.0551**	0.0348
	Improve	109.51%	419.81%	400.91%	383.33%	13.37%	12.91%		58.33%
	R@5	0.0141	0.0035	0.0037	0.0039	0.0218	0.0210	**0.0263**	0.0172
	Improve	86.52%	651.43%	610.81%	574.36%	20.64%	25.24%		52.91%
	P@10	0.0226	0.0115	0.0117	0.0117	0.0472	0.0450	**0.0479**	0.0432
	Improve	111.95%	316.52%	309.40%	309.40%	1.48%	6.44%		10.88%
	R@10	0.0244	0.0079	0.0081	0.0085	0.0424	0.0386	**0.0456**	0.0407
	Improve	86.89%	477.22%	462.96%	436.47%	7.55%	18.13%		12.04%

Table V. Performance Comparisons on the Foursquare dataset with $K = 30$

Ratio	Metrics	Dimension = 30						
		MGM	PMF	PFM	PMFMGM	BPR	BPRLR1	BPRLR2
70%	P@5	0.0409	0.0621	0.0718	0.1201	0.1086	0.1484	**0.1783**
	Improve	335.94%	187.12%	148.33%	48.46%	64.18%	20.15%	
	R@5	0.0306	0.0277	0.0312	0.0594	0.0528	0.0763	**0.0901**
	Improve	194.44%	225.27%	188.78%	51.68%	70.64%	18.09%	
	P@10	0.0373	0.0638	0.0663	0.1166	0.1107	0.1522	**0.1698**
	Improve	355.23%	166.14%	156.11%	45.63%	53.39%	11.56%	
	R@10	0.0531	0.0574	0.0622	0.1166	0.1070	0.1568	**0.1728**
	Improve	225.42%	201.05%	177.81%	48.20%	61.50%	10.20%	
80%	P@5	0.0288	0.0450	0.0482	0.0833	0.0820	0.1050	**0.1287**
	Improve	346.88%	186.00%	167.01%	54.50%	56.95%	22.57%	
	R@5	0.0332	0.0306	0.0364	0.0640	0.0606	0.0834	**0.0998**
	Improve	200.60%	226.14%	174.18%	55.94%	64.69%	19.66%	
	P@10	0.0265	0.0478	0.0512	0.0811	0.0796	0.1053	**0.1227**
	Improve	363.02%	156.69%	139.65%	51.29%	54.15%	16.52%	
	R@10	0.0586	0.0657	0.0677	0.1242	0.1176	0.1658	**0.1898**
	Improve	223.89%	188.89%	180.35%	52.82%	61.39%	14.48%	

Results

Table III. Performance Comparisons on the Gowalla dataset with $K = 30$

Ratio	Metrics	Dimension = 30							
		MGM	PMF	PMFSR	PFM	FMFMGM	BPR	BPRLR1	BPRLR2
70%	P@5	0.0317	0.0148	0.0158	0.0173	0.0672	0.0674	**0.0802**	0.0517
	Improve	153.00%	441.89%	407.59%	363.58%	19.35%	18.99%		55.13%
	R@5	0.0113	0.0033	0.0035	0.0040	0.0212	0.0199	**0.0270**	0.0175
	Improve	138.94%	718.18%	671.43%	575.00%	27.36%	35.68%		54.29%
	P@10	0.0273	0.0162	0.0174	0.0173	0.0656	0.0643	**0.0700**	0.0628
	Improve	156.41%	332.10%	302.30%	304.62%	6.71%	8.86%		11.46%
	R@10	0.0194	0.0075	0.0080	0.0084	0.0408	0.0382	**0.0465**	0.0408
	Improve	260.82%	833.33%	775.00%	733.33%	71.57%	83.25%		71.57%
80%	P@5	0.0263	0.0106	0.011	0.0114	0.0486	0.0488	**0.0551**	0.0348
	Improve	109.51%	419.81%	400.91%	383.33%	13.37%	12.91%		58.33%
	R@5	0.0141	0.0035	0.0037	0.0039	0.0218	0.0210	**0.0263**	0.0172
	Improve	86.52%	651.43%	610.81%	574.36%	20.64%	25.24%		52.91%
	P@10	0.0226	0.0115	0.0117	0.0117	0.0472	0.0450	**0.0479**	0.0432
	Improve	111.95%	316.52%	309.40%	309.40%	1.48%	6.44%		10.88%
	R@10	0.0244	0.0079	0.0081	0.0085	0.0424	0.0386	**0.0456**	0.0407
	Improve	86.89%	477.22%	462.96%	436.47%	7.55%	18.13%		12.04%

Table V. Performance Comparisons on the Foursquare dataset with $K = 30$

Ratio	Metrics	Dimension = 30						
		MGM	PMF	PFM	FMFMGM	BPR	BPRLR1	BPRLR2
70%	P@5	0.0409	0.0621	0.0718	0.1201	0.1086	0.1484	**0.1783**
	Improve	335.94%	187.12%	148.33%	48.46%	64.18%	20.15%	
	R@5	0.0306	0.0277	0.0312	0.0594	0.0528	0.0763	**0.0901**
	Improve	194.44%	225.27%	188.78%	51.68%	70.64%	18.09%	
	P@10	0.0373	0.0638	0.0663	0.1166	0.1107	0.1522	**0.1698**
	Improve	355.23%	166.14%	156.11%	45.63%	53.39%	11.56%	
	R@10	0.0531	0.0574	0.0622	0.1166	0.1070	0.1568	**0.1728**
	Improve	225.42%	201.05%	177.81%	48.20%	61.50%	10.20%	
80%	P@5	0.0288	0.0450	0.0482	0.0833	0.0820	0.1050	**0.1287**
	Improve	346.88%	186.00%	167.01%	54.50%	56.95%	22.57%	
	R@5	0.0332	0.0306	0.0364	0.0640	0.0606	0.0834	**0.0998**
	Improve	200.60%	226.14%	174.18%	55.94%	64.69%	19.66%	
	P@10	0.0265	0.0478	0.0512	0.0811	0.0796	0.1053	**0.1227**
	Improve	363.02%	156.69%	139.65%	51.29%	54.15%	16.52%	
	R@10	0.0586	0.0657	0.0677	0.1242	0.1176	0.1658	**0.1898**
	Improve	223.89%	188.89%	180.35%	52.82%	61.39%	14.48%	

Results

Table III. Performance Comparisons on the Gowalla dataset with $K = 30$

Ratio	Metrics	Dimension = 30							
		MGM	PMF	PMFSR	PFM	FMFMGM	BPR	BPRLR1	BPRLR2
70%	P@5	0.0317	0.0148	0.0158	0.0173	0.0672	0.0674	**0.0802**	0.0517
	Improve	153.00%	441.89%	407.59%	363.58%	19.35%	18.99%		55.13%
	R@5	0.0113	0.0033	0.0035	0.0040	0.0212	0.0199	**0.0270**	0.0175
	Improve	138.94%	718.18%	671.43%	575.00%	27.36%	35.68%		54.29%
	P@10	0.0273	0.0162	0.0174	0.0173	0.0656	0.0643	**0.0700**	0.0628
	Improve	156.41%	332.10%	302.30%	304.62%	6.71%	8.86%		11.46%
	R@10	0.0194	0.0075	0.0080	0.0084	0.0408	0.0382	**0.0465**	0.0408
	Improve	260.82%	833.33%	775.00%	733.33%	71.57%	83.25%		71.57%
80%	P@5	0.0263	0.0106	0.011	0.0114	0.0486	0.0488	**0.0551**	0.0348
	Improve	109.51%	419.81%	400.91%	383.33%	13.37%	12.91%		58.33%
	R@5	0.0141	0.0035	0.0037	0.0039	0.0218	0.0210	**0.0263**	0.0172
	Improve	86.52%	651.43%	610.81%	574.36%	20.64%	25.24%		52.91%
	P@10	0.0226	0.0115	0.0117	0.0117	0.0472	0.0450	**0.0479**	0.0432
	Improve	111.95%	316.52%	309.40%	309.40%	1.48%	6.44%		10.88%
	R@10	0.0244	0.0079	0.0081	0.0085	0.0424	0.0386	**0.0456**	0.0407
	Improve	86.89%	477.22%	462.96%	436.47%	7.55%	18.13%		12.04%

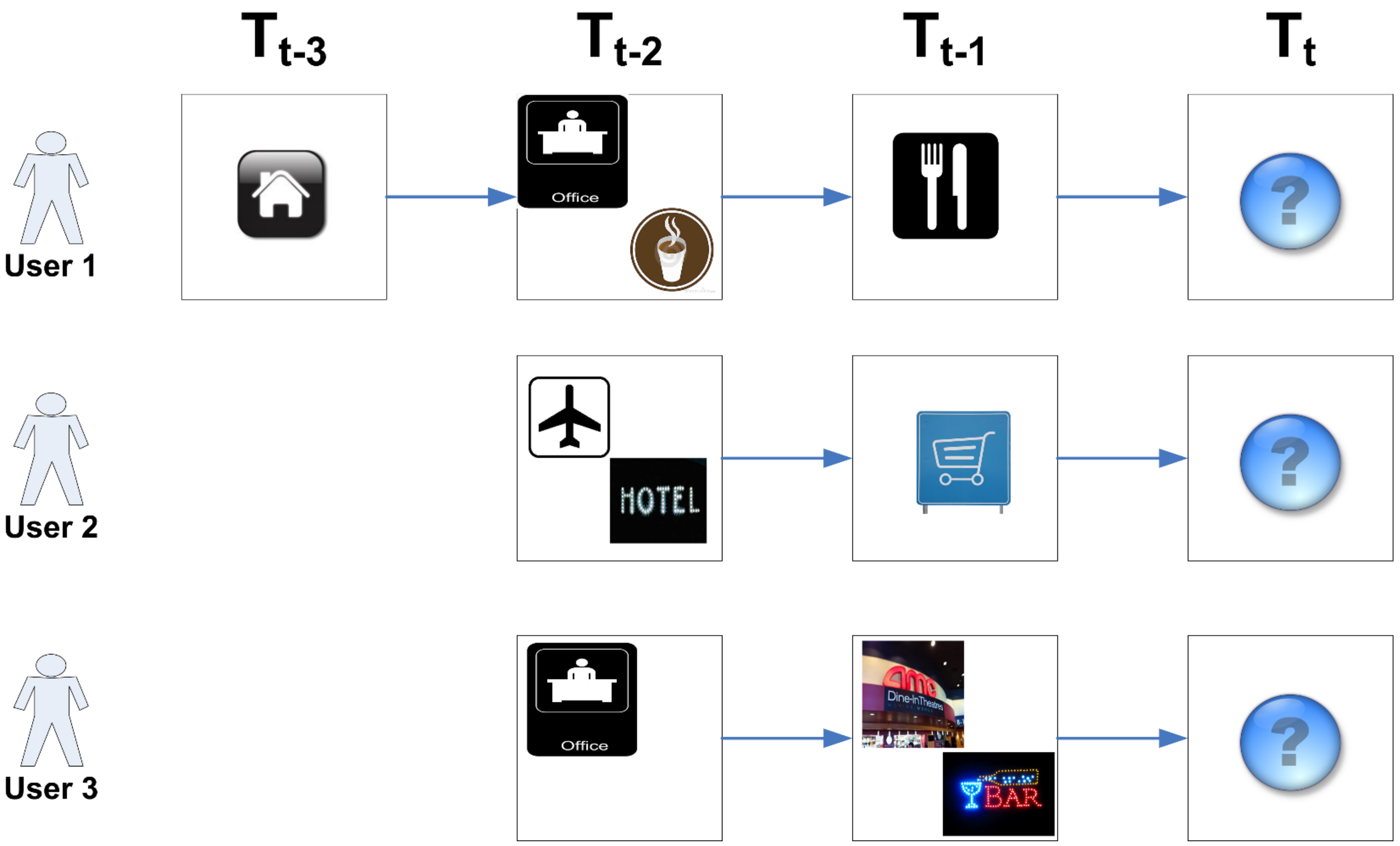
Table V. Performance Comparisons on the Foursquare dataset with $K = 30$

Ratio	Metrics	Dimension = 30						
		MGM	PMF	PFM	FMFMGM	BPR	BPRLR1	BPRLR2
70%	P@5	0.0409	0.0621	0.0718	0.1201	0.1086	0.1484	**0.1783**
	Improve	335.94%	187.12%	148.33%	48.46%	64.18%	20.15%	
	R@5	0.0306	0.0277	0.0312	0.0594	0.0528	0.0763	**0.0901**
	Improve	194.44%	225.27%	188.78%	51.68%	70.64%	18.09%	
	P@10	0.0373	0.0638	0.0663	0.1166	0.1107	0.1522	**0.1698**
	Improve	355.23%	166.14%	156.11%	45.63%	53.39%	11.56%	
	R@10	0.0531	0.0574	0.0622	0.1166	0.1070	0.1568	**0.1728**
	Improve	225.42%	201.05%	177.81%	48.20%	61.50%	10.20%	
80%	P@5	0.0288	0.0450	0.0482	0.0833	0.0820	0.1050	**0.1287**
	Improve	346.88%	186.00%	167.01%	54.50%	56.95%	22.57%	
	R@5	0.0332	0.0306	0.0364	0.0640	0.0606	0.0834	**0.0998**
	Improve	200.60%	226.14%	174.18%	55.94%	64.69%	19.66%	
	P@10	0.0265	0.0478	0.0512	0.0811	0.0796	0.1053	**0.1227**
	Improve	363.02%	156.69%	139.65%	51.29%	54.15%	16.52%	
	R@10	0.0586	0.0657	0.0677	0.1242	0.1176	0.1658	**0.1898**
	Improve	223.89%	188.89%	180.35%	52.82%	61.39%	14.48%	

Outline

- Introduction and Background
- POI Recommendation
- Successive POI Recommendation
- Gradient Boosting Factorization Machines
- Conclusion

Successive POI Recommendation

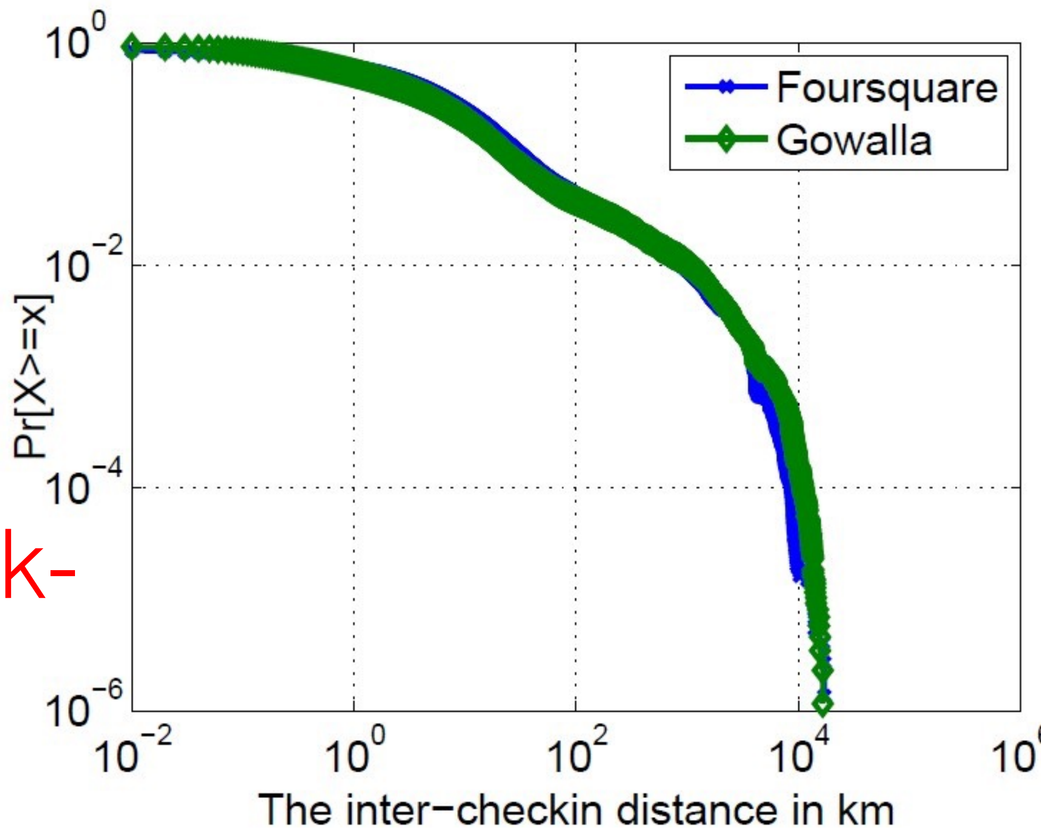


Two Main Properties in LBSNs Dataset

- Personalized Markov chain
- Localized region constraint

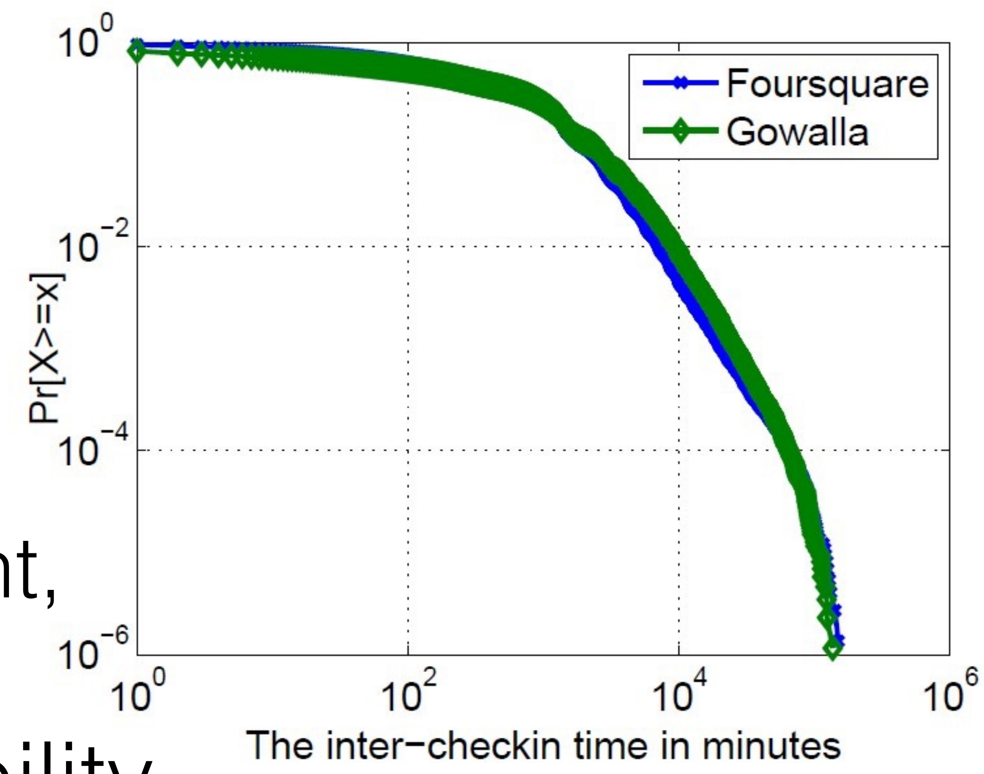
Localized Region Constraint

- Most inter check-ins occurs at nearby locations
  - 75% within 10km, less than 5% beyond 100 km.
- We can only consider the new POIs near a user's previous check-ins when providing successive POI recommendation.



Personalized Markov Chain

- Inter check-in time
  - Around 45% successive check-ins within 2h, 70% within 12h.
- Strong connections between inter check-ins
  - E.g. cinemas or bars after restaurant, hotels after airports.
- Motivated to use transition probability



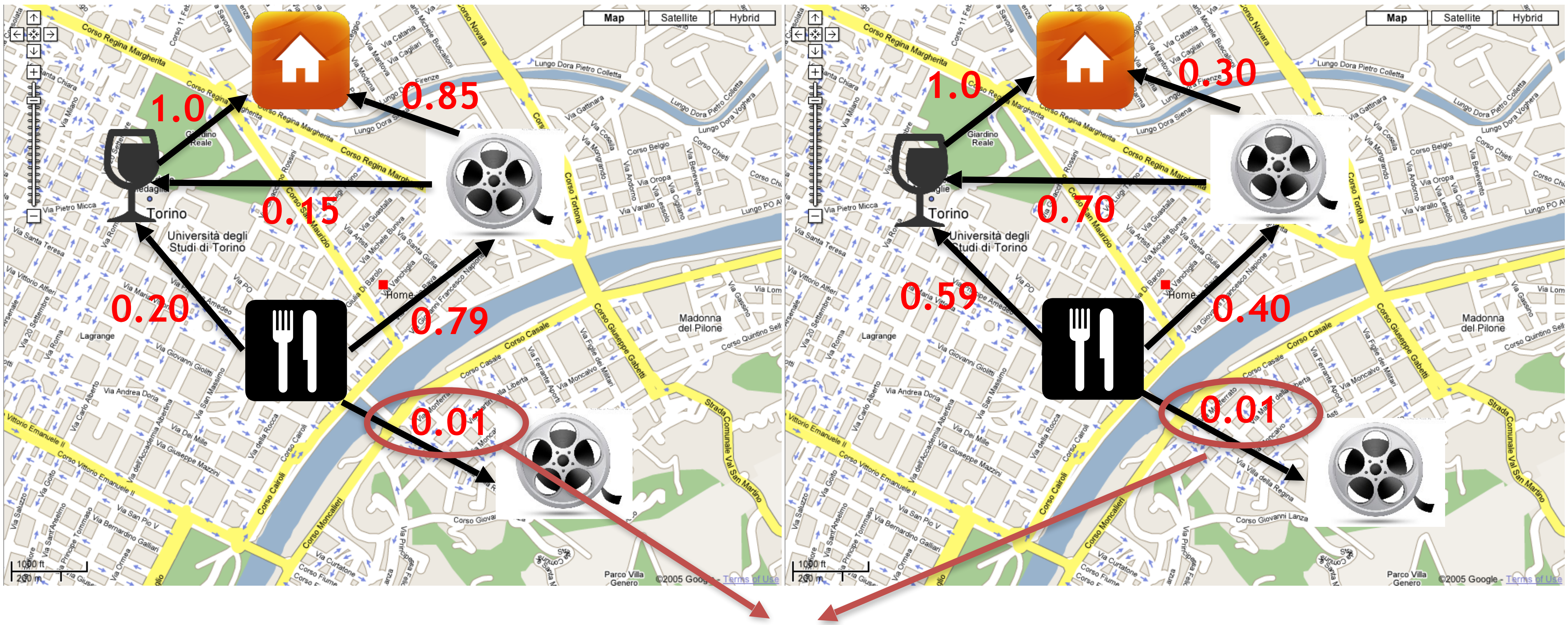
Personalized Markov Chain

- Transition probability: location-wise level or topic level?
  - average user check-in around 50 POIs (Gowalla)
  - 60,000 POIs (Gowalla)
  - location-wise level may be too sparse
  - latent topic level can relieve this problem

Table 4.1: Top 20 topic transitions in Gowalla

Topic(from)	**Topic**(to)
Conference	Home
Tram	Library
Sports	Coffee Shop
Hotel	Mall
Outdoors	Food
Entertainment	Starbucks
Pub	Subway
Golf Shop	Coffee Shop
Hotel	Food
School	Apartment
Movie	Art & Culture
Apparel	Food
Four Seasons	Train Station
Museum	Food
Bears Sports	Mall
Aquatics	Bakery
Rental Car	Coffee Shop
Apparel	Gas & Automotive
Lab	Burgers
Cave	Breakfast

Example



User 1 **Localized Region Constraint** **User 2**

Our Proposal

Our Proposal

- Factoring Personalize Markov Chain with Localized Region model (FPMC-LR)

Our Proposal

- Factoring Personalize Markov Chain with Localized Region model (FPMC-LR)
- Factoring Personalize Markov Chain with Latent Topic Transition (FPMC-LTT)

Our Proposal

- Factoring Personalize Markov Chain with Localized Region model (FPMC-LR)
- Factoring Personalize Markov Chain with Latent Topic Transition (FPMC-LTT)
  - Combine the personalize Markov chain and localized region constraint

Our Proposal

- Factoring Personalize Markov Chain with Localized Region model (FPMC-LR)
- Factoring Personalize Markov Chain with Latent Topic Transition (FPMC-LTT)
  - Combine the personalize Markov chain and localized region constraint
  - Although borrows the idea of FPMC [Rendle et al. '10], we emphasize on users' movement constraint and focus on a different problem

Problem Definition

Problem Definition

- Notation:
  - $\mathcal{U}$: users, $\mathcal{L}$: locations, $\mathcal{L}_u$: the check-in history of user u
  - T: slice window to construct a set check-ins, $\mathcal{T}$: time window set
  - $\mathcal{L}_u^t$: check-in time of user u at time t , $t \in \mathcal{T}$

Problem Definition

- Notation:
  - $\mathcal{U}$: users, $\mathcal{L}$: locations, $\mathcal{L}_u$: the check-in history of user u
  - T: slice window to construct a set check-ins, $\mathcal{T}$: time window set
  - $\mathcal{L}_u^t$: check-in time of user u at time t, $t \in \mathcal{T}$
- Problem:
  - Given a sequence of check-ins, $\mathcal{L}_u^1, \ldots, \mathcal{L}_u^t$, the (lat, lng) pair of locations , recommend POIs to users at t+1

FPMC-LR

FPMC-LR

- FPMC-LR is to recommend a successive personalized POI by the prob. a user u will visit at time t:

$$x_{u,i,l} = p(l \in \mathcal{L}_u^t | i \in \mathcal{L}_u^{t-1})$$

FPMC-LR

- FPMC-LR is to recommend a successive personalized POI by the prob. a user u will visit at time t:

$$x_{u,i,l} = p(l \in \mathcal{L}_u^t | i \in \mathcal{L}_u^{t-1})$$

- Based on first-order Markov chain property

$$p(l \in \mathcal{L}_u^t | \mathcal{L}_u^{t-1}) = \frac{1}{|\mathcal{L}_u^{t-1}|} \sum_{i \in \mathcal{L}_u^{t-1}} p(l \in \mathcal{L}_u^t | i \in \mathcal{L}_u^{t-1})$$

Prob. for user *u* from location *i* to *l*

FPMC-LR

- FPMC-LR only consider the neighbourhood locations of previous check-ins

$$N_d(\mathcal{L}_u^t) = \{l \in \mathcal{L} \setminus \mathcal{L}_u^{t-1} : D(l, l_0) \leq d, \forall l_0 \in \mathcal{L}_u^{t-1}\}$$

- Thus our FPMC-LR yields a transition tensor

$$\mathcal{X} \in [0,1]^{|\mathcal{U}| \times |\mathcal{L}| \times |N_d(\mathcal{L})|}$$

  - Note: $|N_d(\mathcal{L})|$ is reduced largely compared to $|\mathcal{L}|$, around 100 when d = 40 km

FPMC-LR

- Use the same idea in [Rendle et al, '10], we approximate the tensor as:

$$\hat{x}_{u,i,l} = \boldsymbol{v}_u^{\mathcal{U},\mathcal{L}} \cdot \boldsymbol{v}_l^{\mathcal{L},\mathcal{U}} + \boldsymbol{v}_l^{\mathcal{L},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{L}} + \boldsymbol{v}_u^{\mathcal{U},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{U}}$$

- We have:

$$\hat{x}_{u,t,l} = \boldsymbol{v}_u^{\mathcal{U},\mathcal{L}} \cdot \boldsymbol{v}_l^{\mathcal{L},\mathcal{U}} + \frac{1}{|\mathcal{L}_u^{t-1}|} \sum_{i \in \mathcal{L}_u^{t-1}} \boldsymbol{v}_l^{\mathcal{L},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{L}}$$

FPMC-LR

- Use the same idea in [Rendle et al, '10], we approximate the tensor as:

$$\hat{x}_{u,i,l} = \boldsymbol{v}_u^{\mathcal{U},\mathcal{L}} \cdot \boldsymbol{v}_l^{\mathcal{L},\mathcal{U}} + \boldsymbol{v}_l^{\mathcal{L},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{L}} + \boldsymbol{v}_u^{\mathcal{U},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{U}}$$

- We have:

$$\hat{x}_{u,t,l} = \boldsymbol{v}_u^{\mathcal{U},\mathcal{L}} \cdot \boldsymbol{v}_l^{\mathcal{L},\mathcal{U}} + \frac{1}{|\mathcal{L}_u^{t-1}|} \sum_{i \in \mathcal{L}_u^{t-1}} \boldsymbol{v}_l^{\mathcal{L},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{L}}$$

user preference

FPMC-LR

- Use the same idea in [Rendle et al, '10], we approximate the tensor as:

$$\hat{x}_{u,i,l} = \boldsymbol{v}_u^{\mathcal{U},\mathcal{L}} \cdot \boldsymbol{v}_l^{\mathcal{L},\mathcal{U}} + \boldsymbol{v}_l^{\mathcal{L},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{L}} + \boldsymbol{v}_u^{\mathcal{U},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{U}}$$

- We have:

$$\hat{x}_{u,t,l} = \underbrace{\boldsymbol{v}_u^{\mathcal{U},\mathcal{L}} \cdot \boldsymbol{v}_l^{\mathcal{L},\mathcal{U}}}_{\text{user preference}} + \frac{1}{|\mathcal{L}_u^{t-1}|} \sum_{i \in \mathcal{L}_u^{t-1}} \underbrace{\boldsymbol{v}_l^{\mathcal{L},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{L}}}_{\text{location-wise transition}}$$

FPMC-LR

- Model top-k recommendations as a ranking over locations:

$$i >_{u,t} j :\Leftrightarrow \hat{x}_{u,t,i} > \hat{x}_{u,t,j}$$

- The MAP estimator is

$$\arg\max_{\Theta} \sum_{u \in \mathcal{U}} \sum_{\mathcal{L}_u^t \in \mathcal{L}_u} \sum_{i \in \mathcal{L}_u^t} \sum_{j \in N(\mathcal{L}_u^{t-1}) \setminus \mathcal{L}_u^t} \ln \sigma(\hat{x}_{u,t,i} - \hat{x}_{u,t,j}) - \lambda_{\Theta} \|\Theta\|_F^2$$

- Learning algorithm: Stochastic gradient descent

FPMC-LTT

FPMC-LTT

- Maximize similarity between latent vector of location l and the expected average location latent vector after transition

FPMC-LTT

- Maximize similarity between latent vector of location l and the expected average location latent vector after transition

- The probability is:

FPMC-LTT

- Maximize similarity between latent vector of location l and the expected average location latent vector after transition

- The probability is:

global latent topic transition matrix

$$\hat{x}_{u,t,l} = \eta \boldsymbol{U}_u \cdot \boldsymbol{L}_l + (1-\eta) Sim(\boldsymbol{L}_l, \frac{1}{|\mathcal{L}_u^{t-1}|} \boldsymbol{A}^T \sum_{i \in \mathcal{L}_u^{t-1}} \boldsymbol{L}_i)$$

user preference

Dataset

- Two publicly available data sets: Foursquare and Gowalla

Table 4.2: Basic statistics of the Foursquare and Gowalla dataset for successive POI recommendation

	$\#U$	$\#L$	# check-in	# avg. check-in
Foursquare	3571	28754	744055	208.36
Gowalla	4510	59355	873071	193.58

Results

Table 4.3: Performance comparison on Foursquare

Metrics	PMF	PTF	FPMC	FPMC-LR	FPMC-LLT
P@10	0.0185	0.0170	0.0275	0.0360	0.0370
Improve	100.00%	117.65%	34.55%	2.78%	
R@10	0.1542	0.1417	0.2325	0.3033	0.3093
Improve	100.58%	118.28%	33.03%	1.98%	
MAP@10	0.0784	0.0712	0.1265	0.1583	0.1612
Improve	105.61%	126.40%	27.43%	1.83%	

Table 4.4: Performance comparison on Gowalla

Metrics	PMF	PTF	FPMC	FPMC-LR	FPMC-LLT
P@10	0.0130	0.0110	0.0220	0.0310	0.0330
Improve	153.85%	200.00%	50.00%	6.45%	
R@10	0.1040	0.0785	0.1575	0.2116	0.2226
Improve	114.04%	183.57%	41.33%	5.20%	
MAP@10	0.0575	0.0473	0.0853	0.1072	0.1126
Improve	95.83%	138.05%	32.00%	5.04%	

Outline

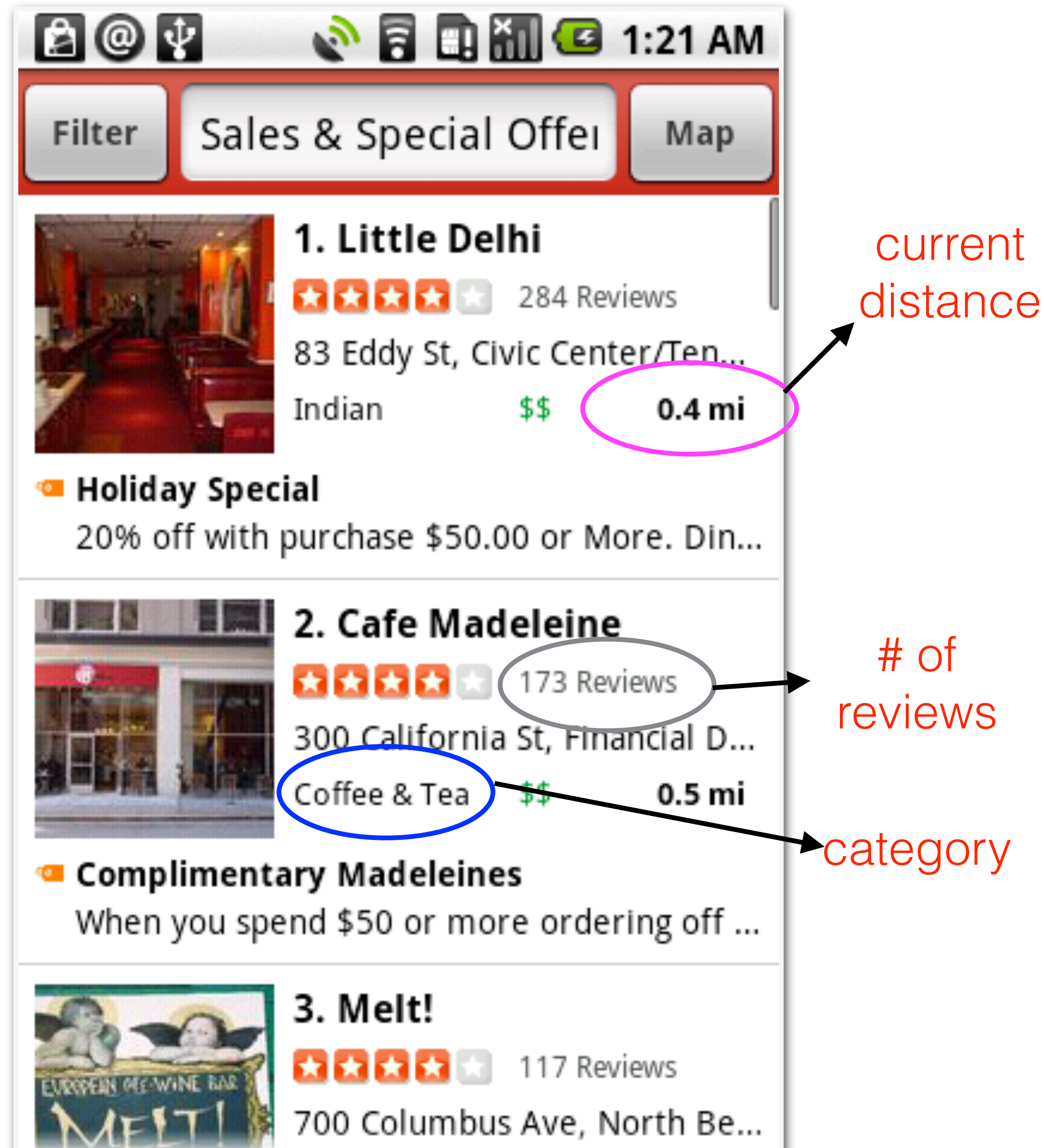
- Introduction and Background
- POI Recommendation
- Successive POI Recommendation
- Gradient Boosting Factorization Machines
- Conclusion

Context-aware Recommendation

- Context features can be helpful
  - User or item meta data: age, genre, etc.
  - Context features attached to the whole event: user's mood, special date, location, etc.

Context-aware Recommendation

- Context features can be helpful
  - User or item meta data: age, genre, etc.
  - Context features attached to the whole event: user's mood, special date, location, etc.



Context-aware Recommendation

- Context features can be helpful
    - User or item meta data: age, genre, etc.
    - Context features attached to the whole event: user's mood, special date, location, etc.

Context-aware Recommendation

- Context features can be helpful
  - User or item meta data: age, genre, etc.
  - Context features attached to the whole event: user’s mood, special date, location, etc.

A Toy Example

$$
\begin{aligned}
\mathcal{U} &= \{u_1, u_2, u_3\} \\
\mathcal{I} &= \{i_1, i_2, i_3, i_4\} \\
\mathcal{M} &= \{Happy, Normal, Sad\}
\end{aligned}
$$

	User			Movie				Mood			…	R
$\mathbf{x}^{(1)}$	1	0	0	1	0	0	0	1	0	0	…	4
$\mathbf{x}^{(2)}$	0	1	0	0	1	0	0	0	0	1	…	2
$\mathbf{x}^{(3)}$	1	0	0	0	1	0	0	0	1	0	…	5
$\mathbf{x}^{(4)}$	0	0	1	0	0	1	0	0	0	1	…	1

User and item are regarded as context features

A Toy Example

$$\begin{aligned}\mathcal{U} &= \{u_1, u_2, u_3\} \\ \mathcal{I} &= \{i_1, i_2, i_3, i_4\} \\ \mathcal{M} &= \{Happy, Normal, Sad\}\end{aligned}$$

User u_1 watched movie i_1 in *Happy* Mood gave rating 4

	User			Movie				Mood			…	R
$\mathbf{x}^{(1)}$	1	0	0	1	0	0	0	1	0	0	…	4
$\mathbf{x}^{(2)}$	0	1	0	0	1	0	0	0	0	1	…	2
$\mathbf{x}^{(3)}$	1	0	0	0	1	0	0	0	1	0	…	5
$\mathbf{x}^{(4)}$	0	0	1	0	0	1	0	0	0	1	…	1

User and item are regarded as context features

Context-aware Factorization Machines

Context-aware Factorization Machines

- A strong baseline proposed in [Rendle et al., 2011.]

Context-aware Factorization Machines

- A strong baseline proposed in [Rendle et al., 2011.]

- Model all interactions between pairs of variables, the rating function is: $\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \hat{w}_{i,j} x_i x_j$

Context-aware Factorization Machines

- A strong baseline proposed in [Rendle et al., 2011.]
- Model all interactions between pairs of variables, the rating function is: $\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \hat{w}_{i,j} x_i x_j$

all pairwise feature interactions

Context-aware Factorization Machines

- A strong baseline proposed in [Rendle et al., 2011.]

- Model all interactions between pairs of variables, the rating function is: $\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \hat{w}_{i,j} x_i x_j$
    - where $\hat{w}_{i,j} := \langle \mathbf{v}_i, \mathbf{v}_j \rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}.$

all pairwise feature interactions

Context-aware Factorization Machines

- A strong baseline proposed in [Rendle et al., 2011.]

- Model all interactions between pairs of variables, the rating function is: $\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \hat{w}_{i,j} x_i x_j$
  - where $\hat{w}_{i,j} := \langle \mathbf{v}_i, \mathbf{v}_j \rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}.$

all pairwise feature interactions

low rank latent feature vector, shared among interacting features
e.g. latent vector U is shared in <U,I> and <U,M>

Drawbacks of FM

Drawbacks of FM

- All interacting features are useful? Or part of them?

Drawbacks of FM

- All interacting features are useful? Or part of them?
    - <U,M>,<U,I>,<I,M> or just <I,M>, <U,I> is enough

Drawbacks of FM

- All interacting features are useful? Or part of them?
  - <U,M>,<U,I>,<I,M> or just <I,M>, <U,I> is enough
  - Not all feature interactions are useful, shared latent features may introduce noise

Drawbacks of FM

- All interacting features are useful? Or part of them?
    - <U,M>,<U,I>,<I,M> or just <I,M>, <U,I> is enough
    - Not all feature interactions are useful, shared latent features may introduce noise
    - Select useful interacting features from tens of features is important

Our Proposal

Our Proposal

- Propose a greedy interacting feature selection algorithm to select useful feature step by step using gradient boosting

Our Proposal

- Propose a greedy interacting feature selection algorithm to select useful feature step by step using gradient boosting

- Propose Gradient Boosting Factorization Machines to incorporate interacting feature selection algorithm and factorization machines into a unified framework

Gradient Boosting Factorization Machines

Gradient Boosting Factorization Machines

- We update the prediction function step by step after selecting interacting features C_p and C_q at step s:

$$\hat{y}_s(\mathbf{x}) := \hat{y}_{s-1}(\mathbf{x}) + \sum_{i \in \mathcal{C}_p} \sum_{j \in \mathcal{C}_q} \mathbb{I}[i, j \in \mathbf{x}] \langle \mathbf{V}_p^i, \mathbf{V}_q^j \rangle$$

Gradient Boosting Factorization Machines

- We update the prediction function step by step after selecting interacting features C_p and C_q at step s:

$$\hat{y}_s(\mathbf{x}) := \hat{y}_{s-1}(\mathbf{x}) + \sum_{i \in \mathcal{C}_p} \sum_{j \in \mathcal{C}_q} \mathbb{I}[i, j \in \mathbf{x}] \langle \mathbf{V}_p^i, \mathbf{V}_q^j \rangle$$

has feature value i in feature C_p
and feature value j in feature C_q

Gradient Boosting Factorization Machines

- We update the prediction function step by step after selecting interacting features C_p and C_q at step s:

$$\hat{y}_s(\mathbf{x}) := \hat{y}_{s-1}(\mathbf{x}) + \sum_{i \in \mathcal{C}_p} \sum_{j \in \mathcal{C}_q} \mathbb{I}[i, j \in \mathbf{x}] \langle \mathbf{V}_p^i, \mathbf{V}_q^j \rangle$$

has feature value i in feature C_p and feature value j in feature C_q

latent feature matrices for feature C_p and C_q to be estimated, usually by stochastic gradient descent (SGD)

Gradient Boosting Factorization Machines

Algorithm 1 Gradient Boosting Factorization Machines Model

Input: Training Data $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$

Output: $\hat{y}_S(x) = \hat{y}_0(x) + \sum_{s=1}^{S} \langle \mathbf{v}_{si}, \mathbf{v}_{sj} \rangle$

Initialize rating prediction function as $\hat{y}_0(x)$

for $s = 1 \rightarrow S$ **do**

  Select interaction feature C_p and C_q from *Greedy Feature Selection Algorithm*

  Estimate latent feature matrices $\mathbf{V}_p$ and $\mathbf{V}_q$

  Update $\hat{y}_s(\mathbf{x}) := \hat{y}_{s-1}(\mathbf{x}) + \sum_{i \in \mathcal{C}_p} \sum_{j \in \mathcal{C}_q} \mathbb{I}[i, j \in \mathbf{x}] \langle \mathbf{V}_p^i, \mathbf{V}_q^j \rangle$

end for

Greedy Feature Selection Algorithm

Greedy Feature Selection Algorithm

- Search a function f that minimizes the objective function:

$$\mathcal{L} = \sum_{i=1}^{N} l(\hat{y}_s(\mathbf{x}_i), y_i) + \Omega(f)$$

- where $\hat{y}_s(\mathbf{x}) = \hat{y}_{s-1}(\mathbf{x}) + f_s(\mathbf{x})$

Greedy Feature Selection Algorithm

- Search a function f that minimizes the objective function:

$$\mathcal{L} = \sum_{i=1}^{N} l(\hat{y}_s(\mathbf{x}_i), y_i) + \Omega(f)$$

N: number of training samples

- where $\hat{y}_s(\mathbf{x}) = \hat{y}_{s-1}(\mathbf{x}) + f_s(\mathbf{x})$

Greedy Feature Selection Algorithm

- Search a function f that minimizes the objective function:

$$\mathcal{L} = \sum_{i=1}^{N} l(\hat{y}_s(\mathbf{x}_i), y_i) + \Omega(f)$$

- where $\hat{y}_s(\mathbf{x}) = \hat{y}_{s-1}(\mathbf{x}) + f_s(\mathbf{x})$

Greedy Feature Selection Algorithm

- Search a function f that minimizes the objective function:

$$\mathcal{L} = \sum_{i=1}^{N} l(\hat{y}_s(\mathbf{x}_i), y_i) + \Omega(f)$$

  - where $\hat{y}_s(\mathbf{x}) = \hat{y}_{s-1}(\mathbf{x}) + f_s(\mathbf{x})$

- We heuristically select feature layer by layer, feasible to compute, suppose feature $C_{i(t)}$ is selected at layer t:

$$f_t(\mathbf{x}) = f_{t-1}(\mathbf{x}) \cdot q_{\mathcal{C}_{i(t)}}(\mathbf{x})$$

Greedy Feature Selection Algorithm

- Search a function f that minimizes the objective function:

$$\mathcal{L} = \sum_{i=1}^{N} l(\hat{y}_s(\mathbf{x}_i), y_i) + \Omega(f)$$

  - where $\hat{y}_s(\mathbf{x}) = \hat{y}_{s-1}(\mathbf{x}) + f_s(\mathbf{x})$

- We heuristically select feature layer by layer, feasible to compute, suppose feature $C_{i(t)}$ is selected at layer t:

$$f_t(\mathbf{x}) = f_{t-1}(\mathbf{x}) \cdot q_{C_{i(t)}}(\mathbf{x})$$

Current layer, in our paper we only consider 2-way interaction, e.g. layer number is 2

Greedy Feature Selection Algorithm

- Search a function f that minimizes the objective function:

$$\mathcal{L} = \sum_{i=1}^{N} l(\hat{y}_s(\mathbf{x}_i), y_i) + \Omega(f)$$

  - where $\hat{y}_s(\mathbf{x}) = \hat{y}_{s-1}(\mathbf{x}) + f_s(\mathbf{x})$

- We heuristically select feature layer by layer, feasible to compute, suppose feature $C_{i(t)}$ is selected at layer t:

$$f_t(\mathbf{x}) = f_{t-1}(\mathbf{x}) \cdot q_{\mathcal{C}_{i(t)}}(\mathbf{x})$$

Greedy Feature Selection Algorithm

- Search a function f that minimizes the objective function:

$$\mathcal{L} = \sum_{i=1}^{N} l(\hat{y}_s(\mathbf{x}_i), y_i) + \Omega(f)$$

  - where $\hat{y}_s(\mathbf{x}) = \hat{y}_{s-1}(\mathbf{x}) + f_s(\mathbf{x})$

- We heuristically select feature layer by layer, feasible to compute, suppose feature $C_{i(t)}$ is selected at layer t:

$$f_t(\mathbf{x}) = f_{t-1}(\mathbf{x}) \cdot q_{\mathcal{C}_{i(t)}}(\mathbf{x})$$

  - The q function is: $q_{\mathcal{C}_{i(t)}}(\mathbf{x}) = \sum_{j \in \mathcal{C}_{i(t)}} \mathbb{I}[j \in \mathbf{x}] \cdot w_{tj}$

Greedy Feature Selection Algorithm

- Search a function f that minimizes the objective function:

$$\mathcal{L} = \sum_{i=1}^{N} l(\hat{y}_s(\mathbf{x}_i), y_i) + \Omega(f)$$

  - where $\hat{y}_s(\mathbf{x}) = \hat{y}_{s-1}(\mathbf{x}) + f_s(\mathbf{x})$

- We heuristically select feature layer by layer, feasible to compute, suppose feature $C_{i(t)}$ is selected at layer t:

$$f_t(\mathbf{x}) = f_{t-1}(\mathbf{x}) \cdot q_{\mathcal{C}_{i(t)}}(\mathbf{x})$$

the corresponding non-zero feature weight suppose choosing feature $C_{i(t)}$ at layer t

  - The q function is: $q_{\mathcal{C}_{i(t)}}(\mathbf{x}) = \boxed{\sum_{j \in \mathcal{C}_{i(t)}} \mathbb{I}[j \in \mathbf{x}] \cdot w_{tj}}$

Greedy Feature Selection Algorithm

Greedy Feature Selection Algorithm

- After n layers, we will select n features, in our paper, we only consider n = 2, which results in a 2-way interacting feature

Greedy Feature Selection Algorithm

- After n layers, we will select n features, in our paper, we only consider n = 2, which results in a 2-way interacting feature
- At layer t, approximated by Taylor expansion the problem is equal to minimize:

Greedy Feature Selection Algorithm

- After n layers, we will select n features, in our paper, we only consider n = 2, which results in a 2-way interacting feature

- At layer t, approximated by Taylor expansion the problem is equal to minimize:

$$\mathcal{L}(f) = \sum_{i=1}^{N} h_i (g_i / h_i - f_t(\mathbf{x}_i))^2 + \Omega(f_t)$$

Greedy Feature Selection Algorithm

- After n layers, we will select n features, in our paper, we only consider n = 2, which results in a 2-way interacting feature

- At layer t, approximated by Taylor expansion the problem is equal to minimize:

$$\mathcal{L}(f) = \sum_{i=1}^{N} h_i(g_i/h_i - f_t(\mathbf{x}_i))^2 + \Omega(f_t)$$

negative first derivative at sample i

second derivative

Greedy Feature Selection Algorithm

Greedy Feature Selection Algorithm

- At layer t, our problem is to select the feature:

$$arg\,min_{i(t)\in\{1,\ldots,m\}}\mathcal{L}(f)$$

Greedy Feature Selection Algorithm

- At layer t, our problem is to select the feature:

$$arg\, min_{i(t) \in \{1,\ldots,m\}} \mathcal{L}(f)$$

- The corresponding weight can be calculated:

$$w_{ij} = \arg\, min_w \sum_{i=1}^{N} h_i(g_i/h_i - f_{t-1}(\mathbf{x}_i) \cdot \mathbb{I}(j \in \mathbf{x}_i) \cdot w)^2 + \lambda w^2$$

Greedy Feature Selection Algorithm

Algorithm 2 Greedy Feature Selection Algorithm

Input: Training Data $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$,context feature set $\mathcal{C}$
Output: n-way interaction feature $\mathcal{C}_{i(1)}, \ldots, \mathcal{C}_{i(n)}$.
for $l = 1 \rightarrow n$ **do**
  $\mathcal{A} = \emptyset$ // $\mathcal{A}$ is the set of context features already selected
  Maintain two vectors $\mathbf{a}$ and $\mathbf{b}$ for all categorical values in $\mathcal{C}$, both initialized to $\mathbf{0}$
  for $(\mathbf{x}_i, y_i)$ in $\mathcal{S}$ **do**
    compute $tempa = z_i h_i f_{t-1}(\mathbf{x}_i)$ and $tempb = h_i(f_{t-1}(\mathbf{x}_i))^2$
    for $j = 1 \rightarrow d$ **do**
      if $\mathbf{x}_{ij}$ is non-zero and not in $\mathcal{A}$ **then**
        add $tempa$ to $\mathbf{a}_j$ and $tempb$ to $\mathbf{b}_j$
      end if
    end for
  end for
  Compute weight for all categorical features in $\mathcal{C} - \mathcal{A}$ according to Eq. 25.
  Select the feature $\mathcal{C}_{i(l)}$ according to Eq. 24.
  Add feature $\mathcal{C}_{i(l)}$ into $\mathcal{A}$
end for

prepared for computing weight

Discussion

Discussion

- Complexity:

Discussion

- Complexity: $\mathcal{O}(SN + kSN)$
  - *S*: boosting steps, *k*: SGD iterations, *N*: training numbers

Discussion

- Complexity: $\mathcal{O}(SN + kSN)$
  - *S*: boosting steps, *k*: SGD iterations, *N*: training numbers
  - linear to training size

Discussion

- Complexity: $\mathcal{O}(SN + kSN)$
  - *S*: boosting steps, *k*: SGD iterations, *N*: training numbers
  - linear to training size
- GBMF-Opt:

Discussion

- Complexity: $\mathcal{O}(SN + kSN)$
    - *S*: boosting steps, *k*: SGD iterations, *N*: training numbers
    - linear to training size
- GBMF-Opt:
    - after GBMF, we have S interacting features

Discussion

- Complexity: $\mathcal{O}(SN + kSN)$
  - *S*: boosting steps, *k*: SGD iterations, *N*: training numbers
  - linear to training size
- GBMF-Opt:
  - after GBMF, we have S interacting features
  - optimize S features globally with shared latent vectors

Discussion

- Complexity: $\mathcal{O}(SN + kSN)$
    - *S*: boosting steps, *k*: SGD iterations, *N*: training numbers
    - linear to training size
- GBMF-Opt:
    - after GBMF, we have S interacting features
    - optimize S features globally with shared latent vectors
    - fewer parameters, better generalization

Dataset

Dataset

- Synthetic data:

Dataset

- Synthetic data:
  - 10 context features

Dataset

- Synthetic data:
  - 10 context features
  - randomly select 5 interacting features to generate 1-5 ratings

Dataset

- Synthetic data:
  - 10 context features
  - randomly select 5 interacting features to generate 1-5 ratings
- Real data: Tencent microblog data

Dataset

- Synthetic data:
  - 10 context features
  - randomly select 5 interacting features to generate 1-5 ratings
- Real data: Tencent microblog data
  - 18 context features: user, item, follower/followee number, tweet time etc.

Dataset

- Synthetic data:
  - 10 context features
  - randomly select 5 interacting features to generate 1-5 ratings
- Real data: Tencent microblog data
  - 18 context features: user, item, follower/followee number, tweet time etc.
  - Sparse, 70% of users in test data not in training data

Dataset

- Synthetic data:
  - 10 context features
  - randomly select 5 interacting features to generate 1-5 ratings
- Real data: Tencent microblog data
  - 18 context features: user, item, follower/followee number, tweet time etc.
  - Sparse, 70% of users in test data not in training data

Table 5.1: Statistics of datasets

Dataset	# Users	#Items	#Observed Entries
Synethic data	1000	1000	16270
Tencent microblog	2.3 M	6095	73 M

Set up and Metrics

Set up and Metrics

- Synthetic data: randomly remove 20% data as test data, the remaining as training

Set up and Metrics

- Synthetic data: randomly remove 20% data as test data, the remaining as training

- Tencent data: split by the time, last 4 weeks as test

Set up and Metrics

- Synthetic data: randomly remove 20% data as test data, the remaining as training
- Tencent data: split by the time, last 4 weeks as test
- Metrics:
  - MAE and RMSE for synthetic data
  - MAP@k for Tecent data

Results

Table 5.2: Results on the synthetic data in RMSE and MAE

Method	RMSE	MAE
PMF	1.9881	1.7650
FM	1.9216	1.6981
GBFM	1.8959	1.6354
GBFM-Opt	1.8611	1.5762

Table 5.3: Results on the Tencent microblog data in MAP

Method	MAP@1	MAP@3	MAP@5
PMF	22.88%	34.50%	37.95%
FM	24.36%	36.77%	40.32%
GBFM	24.62%	37.17%	40.90%
GBFM-Opt	24.66%	37.23%	40.98%

Outline

- Introduction and Background
- POI Recommendation
- Successive POI Recommendation
- Gradient Boosting Factorization Machines
- Conclusion

Conclusion

Conclusion

- POI recommendation

Conclusion

- POI recommendation
  - a framework considers user preference, geographical influence and personalized ranking together

Conclusion

- POI recommendation
  - a framework considers user preference, geographical influence and personalized ranking together
- Successive POI recommendation

Conclusion

- POI recommendation
  - a framework considers user preference, geographical influence and personalized ranking together
- Successive POI recommendation
  - two matrix factorization methods based on personalized Markov chain and region localization

Conclusion

- POI recommendation
  - a framework considers user preference, geographical influence and personalized ranking together
- Successive POI recommendation
  - two matrix factorization methods based on personalized Markov chain and region localization
- Gradient Boosting Factorization Machines

Conclusion

- POI recommendation
  - a framework considers user preference, geographical influence and personalized ranking together
- Successive POI recommendation
  - two matrix factorization methods based on personalized Markov chain and region localization
- Gradient Boosting Factorization Machines
  - incorporate feature selection algorithm with FM

Thanks
Q&A

Set up

- Split the dataset into two non-overlapping sets
  - Randomly select x% for each user as training data and the rest (1-x)% as the test data
  - Carried out 5 times independently, we report the average
- POI recommendation
  - Return top-N POIs for each user
  - Find out # of locations in test dataset are recovered