CSE 258
Web Mining and Recommender Systems

Advanced Recommender Systems
This week

Methodological papers

• Bayesian Personalized Ranking

\[ \sigma \left( \gamma_i \cdot x_i - \gamma_j \cdot x_j \right) \]

• Factorizing Personalized Markov Chains

sequential data

• Personalized Ranking Metric Embedding

\[ \gamma_i \cdot x_i \text{ vs } d(\gamma_i, x_i) \]

• Translation-based Recommendation
This week

Goals:

1. Sequential recsys

2. Different objectives $y_n, x_i \in d(y, i)$
This week

Application papers (Wednesday)

• Recommending Product Sizes to Customers
• Playlist prediction via Metric Embedding
• Efficient Natural Language Response Suggestion for Smart Reply
• Learning Visual Clothing Style with Heterogeneous Dyadic Co-occurrences
This week

We (hopefully?) know enough by now to...
• Read academic papers on Recommender Systems
• Understand most of the models and evaluations used

See also – CSE291
BPR: Bayesian Personalized Ranking from Implicit Feedback

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme
{srendle, freudenthaler, gantner, schmidt-thieme}@ismll.de
Machine Learning Lab, University of Hildesheim
Marienburger Platz 22, 31141 Hildesheim, Germany

Abstract

Item recommendation is the task of predicting a personalized ranking on a set of items (e.g. websites, movies, products). In this paper, we investigate the most common scenario with implicit feedback (e.g. clicks, purchases). There are many methods for item recommendation from implicit feedback like matrix factorization (MF) or adaptive k-nearest-neighbor (kNN). Even though these methods are designed for the item prediction task, they can be easily adapted for the ranking task. However, the resulting ranking is often not personalized enough. In this paper, we propose a new approach for personalized ranking that is based on Bayesian Personalized Ranking (BPR). The idea is to rank the items according to their expected utility for the user. This is achieved by using a probabilistic model that captures the user’s preferences and the item’s relevance. We show that our method outperforms existing approaches in terms of personalization and ranking accuracy.
Bayesian Personalized Ranking

**Goal:** Estimate a personalized *ranking function* for each user

\[ i > u j \]

\[ x(u, i, j) \rightarrow +ve \text{ if } i \text{ preferred by user } u \]

\[ -ve \text{ if } j \text{ preferred by user } u \]
Why? Compare to “traditional” approach of replacing “missing values” by 0:

But! “0”s aren’t necessarily negative!
Bayesian Personalized Ranking

**Why?** Compare to “traditional” approach of replacing “missing values” by 0:

\[ u: \begin{align*}
&+ + + + + \\
&\text{user likes}\end{align*} \quad \begin{align*}
&? ? ? ? ? ? ? \quad \text{doesn’t like} \\
&\Rightarrow \quad \text{doesn’t know about} \]

This suggests a possible solution based on **ranking**
**Bayesian Personalized Ranking**

**Defn:** AUC (for a user $u$)

\[
AUC(u) := \frac{1}{|I_u^+| |I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in |I \setminus I_u^+|} \delta(\hat{x}_{uij} > 0)
\]

The AUC essentially **counts** how many times the model correctly identifies that $u$ prefers the item they bought (positive feedback) over the item they did not.
Bayesian Personalized Ranking

**Defn:** AUC (for a user $u$)

$$AUC(u) := \frac{1}{|I_u^+| |I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in |I \setminus I_u^+|} \delta(\hat{x}_{uij} > 0)$$

**AUC = 1:** We *always* guess correctly among two potential items $i$ and $j$

**AUC = 0.5:** We guess *no better than random*
Bayesian Personalized Ranking

**Defn:** AUC

= Area Under Precision Recall Curve
Bayesian Personalized Ranking

Summary: Goal is to count how many times we identified $i$ as being more preferable than $j$ for a user $u$. $\delta(\hat{x}_{uij} > 0)$
Bayesian Personalized Ranking

**Summary:** Goal is to count how many times we identified \( i \) as being more preferable than \( j \) for a user \( u \)
Bayesian Personalized Ranking

Idea: Replace the counting function $\delta(\hat{x}_{uij} > 0)$ by a smooth function

$$\sigma(\hat{x}_{uij})$$

$\hat{x}_{uij}$ is any function that compares the compatibility of $i$ and $j$ for a user $u$

e.g. could be based on matrix factorization:

$$x_{uij} = \gamma_u \cdot \gamma_i - \gamma_u \cdot \gamma_j$$
Bayesian Personalized Ranking

**Idea:** Replace the counting function $\delta(\hat{x}_{uij} > 0)$ by a smooth function

\[
\text{BPR-Opt} := \ln p(\Theta| >_u)
= \ln p(>_u |\Theta) p(\Theta)
= \ln \prod_{(u,i,j) \in D_S} \sigma(\hat{x}_{uij}) p(\Theta)
= \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta)
= \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda \Theta \|\Theta\|^2
\]
Idea: Replace the counting function $\delta(x_{uij} > 0)$ by a smooth function

$$
\sum_{u,i,j \in T} \ln \sigma(x_{uij} - x_u \cdot x_i \cdot x_j) - \lambda \|x\|^2
$$
Experiments:
• RossMann (online drug store)
• Netflix (treated as a binary problem)
Bayesian Personalized Ranking

Experiments:

Online shopping: Rossmann

Video Rental: Netflix
Morals of the story:

• Given a “one-class” prediction task (like purchase prediction) we might want to optimize a ranking function rather than trying to factorize a matrix directly.

• The AUC is one such measure that counts among a users $u$, items they consumed $i$, and items they did not consume, $j$, how often we correctly guessed that $i$ was preferred by $u$.

• We can optimize this approximately by maximizing $\sigma(\hat{x}_{uij})$ where $\hat{x}_{uij} = \gamma_u \cdot \gamma_i - \gamma_u \cdot \gamma_j$. 
Factorizing Personalized Markov Chains for Next-Basket Recommendation

ABSTRACT

Recommender systems are an important component of many websites. Two of the most popular approaches are based on matrix factorization (MF) and Markov chains (MC). MF methods learn the general taste of a user by factorizing the matrix over observed user-item preferences. On the other hand, MC methods model sequential behavior by learning a transition graph over items that is used to predict the next action based on the recent actions of a user. In this paper, we present a method bringing both approaches together. Our method is based on personalized transition graphs over underlying Markov chains. That means for each user an own transition graph is learned and for new recordings the chain with the highest probability is chosen.

1. INTRODUCTION

A core technology of many recent websites are recommender systems. They are used for example to increase sales in e-commerce, clicking rates on websites or visitor satisfaction in general. In this paper, we deal with the problem setting where sequential basket data is given per user. An obvious example is an online shop where a user buys items (e.g. books or CDs). In these applications, usually several items are bought at the same time, i.e. we have a set/basket of items at one point of time. The target is now to recommend items to the user that he might want to buy in his next visit.
Goal: build temporal models just by looking at the item the user purchased previously

\[ r(u, i|j) \]

(or \( p_u(i|j) \))
Assumption: all of the information contained by temporal models is captured by the previous action. This is what’s known as a **first-order Markov** property.
Is this assumption realistic?

Realistic

Movies

Not realistic

Grocery shopping

Seasonal (e.g. clothing)
Factorizing Personalized Markov Chains for Next-Basket Recommendation

**Data setup:** Rossmann basket data

```
User 1
B_{t-3} → B_{t-2} → B_{t-1} → B_t
```

```
User 2
```

```
User 3
```

```
User 4
```
Factorizing Personalized Markov Chains for Next-Basket Recommendation

**Prediction task:**

\[
p(i \in B_t | B_{t-1}) := \frac{1}{|B_{t-1}|} \sum_{l \in B_{t-1}} p(i \in B_t | l \in B_{t-1})
\]

\[
p(B_t | B_{t-1}) \propto \prod_{i \in B_t} p(i | B_{t-1})
\]
Could we try and compute such probabilities just by counting?

\[
\hat{a}_{l,i} = \hat{p}(i \in B_t | l \in B_{t-1}) = \frac{\hat{p}(i \in B_t \land l \in B_{t-1})}{\hat{p}(l \in B_{t-1})} = \frac{|\{(B_t, B_{t-1}) : i \in B_t \land l \in B_{t-1}\}|}{|\{(B_t, B_{t-1}) : l \in B_{t-1}\}|}
\]

Seems okay, as long as the item vocabulary is small (|I|^2 possible item/item combinations to count)

But it’s not personalized
What if we try to personalize?

$$\hat{a}_{u,l,i} = \hat{p}(i \in B_t^u | l \in B_{t-1}^u) = \frac{\hat{p}(i \in B_t^u \land l \in B_{t-1}^u)}{\hat{p}(l \in B_{t-1}^u)}$$

$$= \frac{|\{(B_t^u, B_{t-1}^u) : i \in B_t^u \land l \in B_{t-1}^u\}|}{|\{(B_t^u, B_{t-1}^u) : l \in B_{t-1}^u\}|}$$

Now we would have U*I^2 counts to compare.

Clearly not feasible, so we need to try and estimate/model this quantity (e.g. by matrix factorization).
What if we try to personalize?

\[ \hat{A} := C \times_U V^U \times_L V^L \times_I V^I \]

\[ C \in \mathbb{R}^{k_U,k_L,k_I}, \quad V^U \in \mathbb{R}^{U \times k_U}, \]
\[ V^L \in \mathbb{R}^{I \times k_L}, \quad V^I \in \mathbb{R}^{I \times k_I} \]
What if we try to personalize?

\[ \hat{a}_{u,i} := \sum_{f=1}^{k_{U,I}} v_{u,f}^{U,I} v_{i,f}^{I,U} + \sum_{f=1}^{k_{I,L}} v_{i,f}^{I,L} v_{l,f}^{L,I} + \sum_{f=1}^{k_{U,L}} v_{u,f}^{U,L} v_{l,f}^{L,U} \]

\[ \theta_u - \theta_i \quad \theta_i - \theta_e \quad \theta_e - \theta_n \]
Factorizing Personalized Markov Chains for Next-Basket Recommendation

Prediction task:

\[ p(i \in B_t | B_{t-1}) := \frac{1}{|B_{t-1}|} \sum_{l \in B_{t-1}} p(i \in B_t | l \in B_{t-1}) \]

\[ \hat{p}(i \in B_t^u | B_{t-1}^u) = \frac{1}{|B_{t-1}^u|} \sum_{l \in B_{t-1}^u} \hat{a}_{u,l,i} \]

\[ = \frac{1}{|B_{t-1}^u|} \sum_{l \in B_{t-1}^u} \left( \langle v_{u}^{U,I} , v_{i}^{I,U} \rangle + \langle v_{i}^{I,L} , v_{l}^{L,I} \rangle + \langle v_{u}^{U,L} , v_{l}^{L,U} \rangle \right) \]
Factorizing Personalized Markov Chains for Next-Basket Recommendation

Prediction task:

$$\arg\max_{\Theta} \prod_{u \in U} \prod_{B_t \in B^u} p(>_{u,t} \mid \Theta) p(\Theta)$$

$$\prod_{u \in U} \prod_{B_t \in B^u} \prod_{i \in B_t} \prod_{j \notin B_t} p(i >_{u,t} j \mid \Theta)$$

$$\sum_{\mathcal{X}} \left( \mathcal{X}_u \cdot \mathcal{X}_i + \mathcal{X}_i \cdot \mathcal{X}_e + \mathcal{X}_u \cdot \mathcal{X}_e - \mathcal{X}_u \cdot \mathcal{X}_j - \mathcal{X}_j \cdot \mathcal{X}_e - \mathcal{X}_u \cdot \mathcal{X}_e \right)$$
Factorizing Personalized Markov Chains for Next-Basket Recommendation

F@5

FMC: not personalized

MF: personalized, but not sequentially-aware
Morals of the story:

• Can improve performance by modeling third order interactions between the user, the item, and the previous item
• This is simpler than temporal models – but makes a big assumption
• Given the blowup in the interaction space, this can be handled by tensor decomposition techniques
Personalized Ranking Metric Embedding for Next New POI Recommendation

Shanshan Feng,1 Xutao Li,2 Yifeng Zeng,3 Gao Cong,2 Yeow Meng Chee,4 Quan Yuan2

1Interdisciplinary Graduate School, Nanyang Technological University, Singapore, sfeng003@e.ntu.edu.sg
2School of Computer Engineering, Nanyang Technological University, Singapore, {lixutao@, gaocong@, quyuan1@e.}ntu.edu.sg
3School of Computing, Teesside University, UK, Y.Zeng@tees.ac.uk
4School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, ymchee@ntu.edu.sg

Abstract

The rapidly growing of Location-based Social Networks (LBSNs) provides a vast amount of check-in data, which enables many services, e.g., point-of-interest (POI) recommendation. In this paper, we study the next new POI recommendation problem in which new POIs with respect to users’ current location are to be recommended. The challenge lies in the difficulty in precisely learning users’ sequential information and personalizing the recommendation model. To this end, we resort to the Metric Embedding method for the recommendation, which avoids drawbacks of the Matrix Factorization technique. We propose a personalized ranking metric embedding method (PRME) to model personalized check-in sequences. We further develop a PRME-G model to learn the sequential behavior of users’ check-ins. The sequential behavior is important for POI recommendation because human movement exhibits sequential patterns [Ye et al., 2013]. We verify users’ sequential behavior in the analysis of two real-world datasets. Meanwhile, we observe that users often visit new POIs that they have not been visited before. In this paper, we focus on the Next New POI recommendation problem (simplified as $N^2$-POI recommendation), which is to recommend new POIs to be visited next given a user’s current location.

The challenge of $N^2$-POI recommendation is to learn transitions of users’ check-ins that are commonly represented by a first-order Markov chain model. Due to the sparse transition data, it is difficult to estimate the transition probability in Markov chain, especially for the unobserved transition. Factorized Personalized Markov Chain (FPMC) [Rendle et al., 2010] method has been used to calculate the item transitions. FPMC exploits matrix factorization technique to factorize the
Goal: Can we build better sequential recommendation models by using the idea of **metric embeddings**

\[ \gamma_u \cdot \gamma_i \text{ vs. } d(\gamma_u, \gamma_i) \]
Why would we expect this to work (or not)?

\[ \hat{Y}_u \cdot \hat{Y}_i \]

\[ d(\hat{Y}_u, \hat{Y}_i) \]

MF, minimizing MSE

metric space

→ geographical

→ playlists
Otherwise, goal is the same as the previous paper:

\[ p_u(i|j) \]
### Personalized Ranking Metric Embedding for Next New POI Recommendation

#### Data

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#User</th>
<th>#POI</th>
<th>#Check-in</th>
<th>Time range</th>
</tr>
</thead>
<tbody>
<tr>
<td>FourSquare</td>
<td>1917</td>
<td>2675</td>
<td>155365</td>
<td>08/2010-07/2011</td>
</tr>
<tr>
<td>Gowalla</td>
<td>4996</td>
<td>6871</td>
<td>245157</td>
<td>11/2009-10/2010</td>
</tr>
</tbody>
</table>
Personalized Ranking Metric Embedding for Next New POI Recommendation

Qualitative analysis

- Sparse
- Cold-start
Personalized Ranking Metric Embedding for Next New POI Recommendation

Qualitative analysis

![Graphs showing Pr(X<x) vs. Hour and Distance (km) for FourSquare and Gowalla.](image-url)
Personalized Ranking Metric Embedding for Next New POI Recommendation

Basic model (not personalized)

\[
\hat{P}(l_j|l_i) = \frac{e^{-||X(l_j) - X(l_i)||^2}}{Z(l_i)}
\]

\[d(i,j) = \prod_\ell X(e_{\ell j}) - X(e_{\ell i})\|_c^2\]
Basic model (not personalized)

\[ l_i >_{l^c} l_j \iff \hat{P}(l_i|l^c) > \hat{P}(l_j|l^c) \]

\[ d(i, c) < d(j, c) \]
Personalized Ranking Metric Embedding for Next New POI Recommendation

Personalized version

\[ D_{u,l,c,l} = \alpha D_{u,l}^P + (1 - \alpha) D_{l,c,l}^S \]
Personalized Ranking Metric Embedding for Next New POI Recommendation

Personalized version

\[ D_{u,l_c,l} = \begin{cases} D_{u,l}^P & \text{if } \Delta(l, l_c) > \tau \\ \alpha D_{u,l}^P + (1 - \alpha) D_{l_c,l}^S & \text{otherwise} \end{cases} \]
Personalized Ranking Metric Embedding for Next New POI Recommendation

**Learning**

\[
P(>_{u,l_c} | \Theta) = P \left( (D_{u,l_c,l_j} - D_{u,l_c,l_i}) > 0 | \Theta \right) \\
= \sigma(D_{u,l_c,l_j} - D_{u,l_c,l_i})
\]
Personalized Ranking Metric Embedding for Next New POI Recommendation

Results

(a) Precision on FourSquare
(b) Recall on FourSquare
(c) Precision on Gowalla
(d) Recall on Gowalla
Morals of the story:

• In some applications, **metric embeddings** might be better than inner products
• Examples could include geographical data, but also others (e.g. playlists?)
Translation-based Recommendation

Ruining He  
UC San Diego  
r4he@cs.ucsd.edu

Wang-Cheng Kang  
UC San Diego  
wckang@eng.ucsd.edu

Julian McAuley  
UC San Diego  
jmcauley@cs.ucsd.edu

ABSTRACT
Modeling the complex interactions between users and items as well as amongst items themselves is at the core of designing successful recommender systems. One classical setting is predicting users’ personalized sequential behavior (or ‘next-item’ recommendation), where the challenges mainly lie in modeling ‘third-order’ interactions between a user, her previously visited item(s), and the next item to consume. Existing methods typically decompose these higher-order interactions into a combination of pairwise relationships, by way of which user preferences (user-item interactions) and sequential patterns (item-item interactions) are captured by separate components. In this paper, we propose a unified method, TransRec, to model such third-order relationships for large-scale sequential prediction. Methodologically, we embed items into a ‘transition space’ where users are modeled as translation vectors operating on item sequences. Empirically, this approach outperforms the state-of-the-art on a wide spectrum of real-world datasets. Data and code are available at https://sites.google.com/a/eng.ucsd.edu/ruining-he/.

1 INTRODUCTION
Modeling and predicting the interactions between users and items, as well as the relationships amongst the items themselves are the main tasks of recommender systems. For instance, in order to predict sequential user actions like the next product to purchase, movie to watch, or place to visit, it is essential (and challenging!) to model the third-order interactions between a user (u), the item(s)
Goal: (e.g.) which movie is this user going to watch next?

Want models that consider
- characteristics/preferences of each user
- local context, i.e., the last consumed item(s)
Translation-based Recommendation

Goal: (e.g) which movie is this user going to watch next?

Viewing history of:

Option 1: Matrix Factorization
Translation-based Recommendation

Goal: (e.g) which movie is this user going to watch next?

viewing history of

Option 2: Markov Chains
**Idea:** Considering the two simultaneously means *modeling the interactions between a user and adjacent items*

\[
\text{Prob}(j \mid u, i)
\]
**Translation-based Recommendation**

**Compare:** Factorized Personalized Markov Chains (earlier today)

\[
Prob(j \mid u, i) \propto \langle \tilde{M}_u, \tilde{N}_j \rangle + \langle \tilde{P}_i, \tilde{Q}_j \rangle
\]

- user preference
- local context
Translation-based Recommendation

**Compare:** Personalized Ranking Metric Embedding (earlier today)

\[ \text{Prob}(j \mid u, i) \propto - \left( \alpha \cdot \| \vec{M}_u - \vec{N}_j \|_2^2 + (1 - \alpha) \cdot \| \vec{P}_i - \vec{P}_j \|_2^2 \right) \]

an additional hyperpara. to balance the two components
**Compare:** Hierarchical Representation Model (HRM)

*Wang et al., 2015*

(earlier today)

\[ Prob(j \mid u, i) \propto \langle \text{aggregation}(\tilde{M}_u, \tilde{N}_i), \tilde{N}_j \rangle \]

average/max pooling, etc.
**Compare:** Hierarchical Representation Model (HRM)  
*Wang et al., 2015*  
(earlier today)

\[
Prob(j \mid u, i) \propto \langle \text{aggregation}(\tilde{M}_u, \tilde{N}_i), \tilde{N}_j \rangle
\]

average/max pooling, etc.

**Goal:** Try and get the “best of both worlds,” by modeling third-order interactions *and* using metric embeddings
**Detour:** Translation models in Knowledge Bases

**Data:** entities; links (multiple types of relationships)

**State-of-the-art method:** ‘relationships as translations’

**Goal:** Predict unseen links

Training example:

Basic idea:

E.g. [Bordes et al., 2013], [Wang et al., 2014], [Lin et al., 2015]
Translation-based Recommendation

Embedding space

\{ Items as points \\
Users as translation vectors \}

Training triplet:

Objective: \[ \vec{\gamma}_i + \vec{t}_u \approx \vec{\gamma}_j \]
Translation-based Recommendation

Embedding space

\{ Items as points

Users as translation vectors

Translation operation:
\text{prev. item} + \text{user} \approx \text{next item}
Translation-based Recommendation

\[
Prob(j|u, i) \propto \beta_j - d(\vec{\gamma}_i + \vec{t}_u, \vec{\gamma}_j)
\]

- Benefit from using metric embeddings
- Model \((u, i, j)\) with a single component
- Recommendations can be made by a simple NN search
\[
\hat{\Theta} = \arg \max_{\Theta} \ln \prod_{u \in \mathcal{U}} \prod_{j \in S^u} \prod_{j' \notin S^u} \text{Prob}(j > u, i, j' | \Theta) \text{Prob}(\Theta) \\
= \arg \max_{\Theta} \sum_{u \in \mathcal{U}} \sum_{j \in S^u} \sum_{j' \notin S^u} \ln \sigma(\hat{p}_{u, i, j} - \hat{p}_{u, i, j'}) - \Omega(\Theta),
\]
Translation-based Recommendation

- Automotives
- Office Products
- Toys & Games
- Video Games
- Cell Phones & Accessories
- Clothing, Shoes, and Jewelry
- Electronics

May 1996 - July 2014
Translation-based Recommendation

**Foursquare**
check-ins at different venues

**Flixster**
movie ratings

**Epinions.com**
user reviews
Jan. 2001 - Nov. 2013
(all available online)
Translation-based Recommendation

11.4M reviews & ratings of 4.5M users on 3.1M local businesses

Characteristics: vast vocabulary of items, variability, and sparsity

http://cseweb.ucsd.edu/~jmcauley/
Translation-based Recommendation

Category Id

Count

Restaurant
Hotel
Bar

Retirement Home

International Airport

#reviews
#businesses
Translation-based Recommendation

| Dataset       | #users ($|U|$) | #items ($|I|$) | #actions   | avg. #actions /user | avg. #actions /item |
|---------------|-------------|---------------|------------|---------------------|---------------------|
| Epinions      | 5,015       | 8,335         | 26,932     | 5.37                | 3.23                |
| Automotive    | 34,316      | 40,287        | 183,573    | 5.35                | 4.56                |
| Google        | 350,811     | 505,516       | 2,591,026  | 7.39                | 5.13                |
| Office        | 16,716      | 22,357        | 128,070    | 7.66                | 5.73                |
| Toys          | 57,617      | 69,147        | 410,920    | 7.13                | 5.94                |
| Clothing      | 184,050     | 174,484       | 1,068,972  | 5.81                | 6.13                |
| Cellphone     | 68,330      | 60,083        | 429,231    | 6.28                | 7.14                |
| Games         | 31,013      | 23,715        | 287,107    | 9.26                | 12.11               |
| Electronics   | 253,996     | 145,199       | 2,109,879  | 8.31                | 14.53               |
| Foursquare    | 43,110      | 13,335        | 306,553    | 7.11                | 22.99               |
| Flixter       | 69,485      | 25,759        | 8,000,971  | 115.15              | 310.61              |

**Total**      | **1.11M**   | **1.09M**     | **15.5M**  |                     |                     |
## Translation-based Recommendation

<table>
<thead>
<tr>
<th>Property</th>
<th>PopRec</th>
<th>BPR-MF</th>
<th>FMC</th>
<th>FPMC</th>
<th>HRM</th>
<th>PRME</th>
<th>TransRec</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>S</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>M</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>U</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Translation-based Recommendation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metric</th>
<th>PopRec</th>
<th>BPR-MF</th>
<th>FMC</th>
<th>FPMC</th>
<th>HRM(_{avg})</th>
<th>HRM(_{max})</th>
<th>PRME</th>
<th>TransRec(_L_1)</th>
<th>TransRec(_L_2)</th>
<th>%Improv.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epinions</td>
<td>AUC</td>
<td>0.4576</td>
<td>0.5523</td>
<td>0.5537</td>
<td>0.5517</td>
<td>0.6060</td>
<td>0.5617</td>
<td>0.6117</td>
<td>0.6063</td>
<td>0.6133</td>
<td>0.3%</td>
</tr>
<tr>
<td></td>
<td>Hit@50</td>
<td>3.42%</td>
<td>3.70%</td>
<td>3.84%</td>
<td>2.93%</td>
<td>3.44%</td>
<td>2.79%</td>
<td>2.51%</td>
<td>3.18%</td>
<td>4.63%</td>
<td>20.6%</td>
</tr>
<tr>
<td>Automotive</td>
<td>AUC</td>
<td>0.5870</td>
<td>0.6342</td>
<td>0.6438</td>
<td>0.6427</td>
<td>0.6704</td>
<td>0.6556</td>
<td>0.6469</td>
<td>0.6779</td>
<td>0.6868</td>
<td>2.5%</td>
</tr>
<tr>
<td></td>
<td>Hit@50</td>
<td>3.84%</td>
<td>3.80%</td>
<td>2.32%</td>
<td>3.11%</td>
<td>4.47%</td>
<td>3.71%</td>
<td>3.42%</td>
<td>5.07%</td>
<td>5.37%</td>
<td>20.1%</td>
</tr>
<tr>
<td>Google</td>
<td>AUC</td>
<td>0.5391</td>
<td>0.8188</td>
<td>0.7619</td>
<td>0.7740</td>
<td>0.8640</td>
<td>0.8102</td>
<td>0.8252</td>
<td>0.8359</td>
<td>0.8691</td>
<td>0.6%</td>
</tr>
<tr>
<td></td>
<td>Hit@50</td>
<td>0.32%</td>
<td>4.27%</td>
<td>3.54%</td>
<td>3.99%</td>
<td>3.55%</td>
<td>4.59%</td>
<td>5.07%</td>
<td>6.37%</td>
<td>6.84%</td>
<td>32.5%</td>
</tr>
<tr>
<td>Office</td>
<td>AUC</td>
<td>0.6427</td>
<td>0.6979</td>
<td>0.6867</td>
<td>0.6866</td>
<td>0.6981</td>
<td>0.7005</td>
<td>0.7020</td>
<td>0.7186</td>
<td>0.7302</td>
<td>4.0%</td>
</tr>
<tr>
<td></td>
<td>Hit@50</td>
<td>1.66%</td>
<td>4.09%</td>
<td>2.66%</td>
<td>2.97%</td>
<td>5.50%</td>
<td>4.17%</td>
<td>6.20%</td>
<td>6.86%</td>
<td>6.51%</td>
<td>10.7%</td>
</tr>
<tr>
<td>Toys</td>
<td>AUC</td>
<td>0.6240</td>
<td>0.7232</td>
<td>0.6645</td>
<td>0.7194</td>
<td>0.7579</td>
<td>0.7258</td>
<td>0.7261</td>
<td>0.7442</td>
<td>0.7590</td>
<td>0.2%</td>
</tr>
<tr>
<td></td>
<td>Hit@50</td>
<td>1.69%</td>
<td>3.60%</td>
<td>1.55%</td>
<td>4.41%</td>
<td>5.25%</td>
<td>3.74%</td>
<td>4.80%</td>
<td>5.46%</td>
<td>5.44%</td>
<td>4.0%</td>
</tr>
<tr>
<td>Clothing</td>
<td>AUC</td>
<td>0.6189</td>
<td>0.6508</td>
<td>0.6640</td>
<td>0.6646</td>
<td>0.7057</td>
<td>0.6862</td>
<td>0.6886</td>
<td>0.7047</td>
<td>0.7243</td>
<td>2.6%</td>
</tr>
<tr>
<td></td>
<td>Hit@50</td>
<td>1.11%</td>
<td>1.05%</td>
<td>0.57%</td>
<td>0.51%</td>
<td>1.70%</td>
<td>1.15%</td>
<td>1.00%</td>
<td>1.76%</td>
<td>2.12%</td>
<td>24.7%</td>
</tr>
<tr>
<td>Cellphone</td>
<td>AUC</td>
<td>0.6959</td>
<td>0.7569</td>
<td>0.7347</td>
<td>0.7375</td>
<td>0.7892</td>
<td>0.7654</td>
<td>0.7860</td>
<td>0.7988</td>
<td>0.8104</td>
<td>2.7%</td>
</tr>
<tr>
<td></td>
<td>Hit@50</td>
<td>4.43%</td>
<td>5.15%</td>
<td>3.23%</td>
<td>2.81%</td>
<td>8.77%</td>
<td>6.32%</td>
<td>6.95%</td>
<td>9.46%</td>
<td>9.54%</td>
<td>8.8%</td>
</tr>
<tr>
<td>Games</td>
<td>AUC</td>
<td>0.7495</td>
<td>0.8517</td>
<td>0.8407</td>
<td>0.8523</td>
<td>0.8776</td>
<td>0.8566</td>
<td>0.8597</td>
<td>0.8711</td>
<td>0.8815</td>
<td>0.4%</td>
</tr>
<tr>
<td></td>
<td>Hit@50</td>
<td>5.17%</td>
<td>10.93%</td>
<td>13.93%</td>
<td>12.29%</td>
<td>14.44%</td>
<td>12.86%</td>
<td>14.22%</td>
<td>16.61%</td>
<td>16.44%</td>
<td>15.0%</td>
</tr>
<tr>
<td>Electronics</td>
<td>AUC</td>
<td>0.7837</td>
<td>0.8096</td>
<td>0.8158</td>
<td>0.8082</td>
<td>0.8212</td>
<td>0.8148</td>
<td>0.8337</td>
<td>0.8457</td>
<td>0.8484</td>
<td>1.8%</td>
</tr>
<tr>
<td></td>
<td>Hit@50</td>
<td>4.62%</td>
<td>2.98%</td>
<td>4.15%</td>
<td>2.82%</td>
<td>4.09%</td>
<td>2.59%</td>
<td>3.07%</td>
<td>4.89%</td>
<td>5.19%</td>
<td>12.3%</td>
</tr>
<tr>
<td>Foursquare</td>
<td>AUC</td>
<td>0.9168</td>
<td>0.9511</td>
<td>0.9463</td>
<td>0.9479</td>
<td>0.9559</td>
<td>0.9523</td>
<td>0.9565</td>
<td>0.9631</td>
<td>0.9651</td>
<td>0.9%</td>
</tr>
<tr>
<td></td>
<td>Hit@50</td>
<td>55.60%</td>
<td>60.03%</td>
<td>63.00%</td>
<td>64.53%</td>
<td>60.75%</td>
<td>61.60%</td>
<td>65.32%</td>
<td>66.12%</td>
<td>67.09%</td>
<td>2.7%</td>
</tr>
<tr>
<td>Flixtor</td>
<td>AUC</td>
<td>0.9459</td>
<td>0.9722</td>
<td>0.9568</td>
<td>0.9718</td>
<td>0.9695</td>
<td>0.9687</td>
<td>0.9728</td>
<td>0.9727</td>
<td>0.9750</td>
<td>0.2%</td>
</tr>
<tr>
<td></td>
<td>Hit@50</td>
<td>11.92%</td>
<td>21.58%</td>
<td>22.23%</td>
<td>33.11%</td>
<td>32.34%</td>
<td>30.88%</td>
<td>40.81%</td>
<td>35.52%</td>
<td>35.02%</td>
<td>-13.0%</td>
</tr>
</tbody>
</table>
Translation-based Recommendation

Works well with...

Doesn’t work well with...
Morals of the story:

• Today we looked at two main ideas that extend the recommender systems we saw in class:

  1. **Sequential Recommendation:** Most of the dynamics due to time can be captured purely by knowing the *sequence* of items

  2. **Metric Recommendation:** In some settings, using inner products may not be the correct assumption
Assignment 1

\[ V_{\text{visit}} \]

\[ \ln \Omega (\chi_n \cdot \chi_i - \chi_n \cdot \chi_j) \]

\[ \phi (\chi_n, i) = \begin{cases} 1, & \text{Jaccard}(i, \text{previous bsz, united by n}) \end{cases} \]
Assignment 1

\[ R_{\theta}g = x + \beta_n + \beta_i + \delta_n \cdot \gamma_i \]

\[ \| \beta_n \| + \| \beta_i \| + \| \beta \| \]

\[ \phi(u_{ii}) = \left[ 1, \alpha + \beta_n + \phi_i + \delta_n \cdot \gamma_i \right] \]
CSE 258
Web Mining and Recommender Systems

Real-world applications of recommender systems
Recommending Product Sizes to Customers

Vivek Sembium
Amazon Development Center India
viveksem@amazon.com

Atul Saroop
Amazon Development Center India
asaroop@amazon.com

Rajeev Rastogi
Amazon Development Center India
rastogi@amazon.com

Srujana Merugu*
srujana@gmail.com

ABSTRACT
We propose a novel latent factor model for recommending product size fits \{Small, Fit, Large\} to customers. Latent factors for customers and products in our model correspond to their physical true size, and are learnt from past product purchase and returns data. The outcome for a customer, product pair is predicted based on the difference between customer and product true sizes, and efficient algorithms are proposed for computing customer and product true size values that minimize two loss function variants. In experiments with Amazon shoe datasets, we show that our latent factor models incorporating personas, and leveraging return codes show a 17-21% AUC improvement compared to baselines. In an online A/B test, our algorithms show an improvement of 0.49% in percentage of Fit transactions over control.

CCS CONCEPTS

In the size recommendation problem, a customer implicitly provides the context of a desired product by viewing the detail page of a product and requires a recommendation for the appropriate size variant of the product. For example, the customer might be viewing the detail page of Nike Women’s Tennis Classic shoe and needs to choose from 10 different size variants corresponding to sizes from 6 to 15. Thus, given the context of a desired product, our objective is to recommend the appropriate size variant for a customer.

The problem of recommending sizes to customers is challenging due to the following reasons:

- Data sparsity. Typically, a small fraction of customers and products account for the bulk of purchases. A majority of customers and products have very few purchases.
- Cold start. The environment is highly dynamic with new customers and products (that have no past purchases) for
**Goal:** Build a recommender system that predicts whether an item will “fit”:

\[(u, i) \rightarrow \{\text{small, fit, large}\}\]
Challenges:

• **Data sparsity**: people have very few purchases from which to estimate size

• **Cold-start**: How to handle new customers and products with no past purchases?

• **Multiple personas**: Several customers may use the same account
Recommending product sizes to customers

**Data:**

- Shoe transactions from Amazon.com
- For each shoe $j$, we have a reported size $c_j$ (from the manufacturer), but this may not be correct!
- Need to estimate the customer’s size ($s_i$), as well as the product’s **true** size ($t_j$)
Recommending product sizes to customers

**Loss function:**

\[
f_w(s_i, t_j) + b
\]

\[
f_w(s_i, t_j) = w \cdot (s_i - t_j)
\]
Recommending product sizes to customers

Loss function:

\[
L(y_{ij}, f_w(s_i, t_j)) = \begin{cases} 
L_{\text{bin}}^{\text{+1}}(f_w(s_i, t_j) - b_2) & \text{if } y_{ij} = \text{Small} \\
L_{\text{bin}}^{\text{-1}}(f_w(s_i, t_j) - b_2) + L_{\text{bin}}^{\text{+1}}(f_w(s_i, t_j) - b_1) & \text{if } y_{ij} = \text{Fit} \\
L_{\text{bin}}^{\text{-1}}(f_w(s_i, t_j) - b_1) & \text{if } y_{ij} = \text{Large}
\end{cases}
\]

\[
L(y_{ij}, f_w(s_i, t_j)) = \begin{cases} 
\log\left(\frac{1}{1 + e^{f_w(s_i, t_j) - b_2}}\right) & \text{if } y_{ij} = \text{Small} \\
\log\left(\frac{1}{1 + e^{f_w(s_i, t_j) - b_2}}\right) + \log\left(\frac{1}{1 + e^{-f_w(s_i, t_j) + b_1}}\right) & \text{if } y_{ij} = \text{Fit} \\
\log\left(\frac{1}{1 + e^{f_w(s_i, t_j) - b_1}}\right) & \text{if } y_{ij} = \text{Large}
\end{cases}
\]
Recommending product sizes to customers

Loss function:

\[
L(y_{ij}, f_w(s_i, t_j)) = \begin{cases} 
\max\{0, 1 - f_w(s_i, t_j) + b_2\} & \text{if } y_{ij} = \text{Small} \\
\max\{0, 1 + f_w(s_i, t_j) - b_2\} & \text{if } y_{ij} = \text{Fit} \\
\max\{0, 1 - f_w(s_i, t_j) + b_1\} & \text{if } y_{ij} = \text{Large}
\end{cases}
\]
Recommending product sizes to customers

Figure 1: Hinge loss value for a Fit transaction vs $s_i$.

Figure 2: Hinge loss value for a Small transaction vs $s_i$.

Figure 3: Hinge loss value for a Large transaction vs $s_i$.

Figure 4: Illustrative overall hinge loss vs $s_i$. 
Recommending product sizes to customers

**Loss function:**

\[
L_i = \sum_{(i,j,y_{ij}) \in D \land y_{ij} = \text{Small}} \max\{0, 1 - f_w(s_i, t_j) + b_2\} \\
+ \sum_{(i,j,y_{ij}) \in D \land y_{ij} = \text{Fit}} \left(\max\{0, 1 + f_w(s_i, t_j) - b_2\} + \max\{0, 1 - f_w(s_i, t_j) + b_1\}\right) \\
+ \sum_{(i,j,y_{ij}) \in D \land y_{ij} = \text{Large}} \max\{0, 1 + f_w(s_i, t_j) - b_1\}
\]
Recommending product sizes to customers

Model fitting:

\[ t_j = c_j \]

1. Fix \( t_j \), update \( s_i \)
2. Fix \( s_i \), update \( t_j \)

repeat until converged
Recommending product sizes to customers

Extensions:

- **Multi-dimensional sizes**
  \[ w_1(s_{i1} - t_{j1}) + w_2(s_{i2} - t_{j2}) \]

- **Customer and product features**
  \[ w(s_i - t_j) + \phi(x, i)w' \]

- **User personas**

  - Clustering on transactions
  - By each cluster is one user
### Recommending product sizes to customers

#### Experiments:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Baseline RF</th>
<th>Baseline Linear</th>
<th>Baseline RF</th>
<th>Algorithm 1 Linear</th>
<th>Algorithm 1 RF</th>
<th>Algorithm 1 Persona RF</th>
<th>Algorithm 1 Persona Linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.6%</td>
<td>0%</td>
<td>0%</td>
<td>16.4%</td>
<td>17.2%</td>
<td>17.9%</td>
<td>18.1%</td>
</tr>
<tr>
<td>B</td>
<td>2.1%</td>
<td>0.4%</td>
<td>2.1%</td>
<td>20.3%</td>
<td>21.3%</td>
<td>21.3%</td>
<td>20.5%</td>
</tr>
<tr>
<td>C</td>
<td>3.8%</td>
<td>1.9%</td>
<td>1.9%</td>
<td>15.7%</td>
<td>16.1%</td>
<td>18.4%</td>
<td>17.8%</td>
</tr>
<tr>
<td>D</td>
<td>2.7%</td>
<td>1.2%</td>
<td>1.6%</td>
<td>20.0%</td>
<td>20.2%</td>
<td>21.3%</td>
<td>20.7%</td>
</tr>
<tr>
<td>E</td>
<td>1.5%</td>
<td>0.2%</td>
<td>0.6%</td>
<td>15.8%</td>
<td>15.6%</td>
<td>17.4%</td>
<td>17.4%</td>
</tr>
<tr>
<td>F</td>
<td>2.5%</td>
<td>2.7%</td>
<td>2.3%</td>
<td>18.1%</td>
<td>17.3%</td>
<td>18.5%</td>
<td>17.3%</td>
</tr>
</tbody>
</table>
Recommending product sizes to customers

Experiments:

*Online A/B test*
Recommendating product sizes to customers

**Morals of the story:**

- Very simple model that actually works well in production
- Only a single parameter per user and per item!
Playlist prediction via Metric Embedding

\[ \mathcal{L}(\mathbf{x}_i, \mathbf{x}_j) \sim \mathbf{x}_i \cdot \mathbf{x}_j \]

ABSTRACT

Digital storage of personal music collections and cloud-based music services (e.g., Pandora, Spotify) have fundamentally changed how music is consumed. In particular, automatically generated playlists have become an important mode of accessing large music collections. The key goal of automated playlist generation is to provide the user with a coherent listening experience. In this paper, we present Latent Markov Embedding (LME), a machine learning algorithm for generating such playlists. In analogy to matrix factorization methods for collaborative filtering, the algorithm does not require songs to be described by features a priori, but it learns a representation from example playlists. We formulate this problem as a regularized maximum-likelihood embedding of Markov chains in Euclidean space, and show how addition, when using a cloud-based service like Rhapsody or Spotify, the consumer has instant on-demand access to millions of songs. This has created substantial interest in automatic playlist algorithms that can help consumers explore large collections of music. Companies like Apple and Pandora have developed successful commercial playlist algorithms, but relatively little is known about how these algorithms work and how well they perform in rigorous evaluations.

Despite the large commercial demand, comparably little scholarly work has been done on automated methods for playlist generation (e.g., [13, 4, 9, 11]), and the results to date indicate that it is far from trivial to operationally define what makes a playlist coherent. The most comprehensive study was done by [11]. Working under a model where each part of the list is defined by a Markov chain with transition...
Goal: Build a recommender system that recommends sequences of songs

Idea: Might also use a metric embedding (consecutive songs should be “nearby” in some space)
Basic model:

$$Pr(p[i]|p[i-1]) = \frac{e^{-||X(p[i]) - X(p[i-1])||_2^2}}{\sum_{j=1}^{S} e^{-||X(s_j) - X(p[i-1])||_2^2}}$$

(compare with metric model from last lecture)

$$\mathcal{A}(p^{(i)}, p^{(i-1)}) = ||X(p^{(i)}) - X(p^{(i-1)})||$$
Playlist prediction via Metric Embedding

Basic model ("single point"): 

[Image of a diagram showing a spiral with an arrow pointing upwards and a cross at the bottom left.]
Playlist prediction via Metric Embedding

“Dual-point” model

\[ \|X(s) - X(s')\|_2 \]

\[ \|V(s) - U(s')\|_2 \]

“end” pos of s

“Start” pos of s'
Extensions:

• Popularity biases

\[ \Pr(p^i|p^{i-1}) = \frac{e^{-\Delta(p^i, p^{i-1})^2 + b_i}}{\sum_j e^{-\Delta(s_j, p^{i-1})^2 + b_j}} \]

More popular → higher probability
Playlist prediction via Metric Embedding

Extensions:

- Personalization

\[
Pr(p^{[i]}|p^{[i-1]}, u) = \frac{e^{-\Delta(p^{[i]}, p^{[i-1]})^2 + A(p^{[i]})^T B(u)}}{\sum_j e^{-\Delta(s_j, p^{[i-1]})^2 + A(s_j)^T B(u)}}
\]
Playlist prediction via Metric Embedding

Extensions:

- Semantic Tags

\[
\Pr(X(s)|T(s)) = \mathcal{N}\left(\frac{1}{|T(s)|} \sum_{t \in T(s)} M(t), \frac{1}{2\lambda} I_d\right)
\]
Extensions:

- Observable Features

\[
\Pr(p[i]|p[i-1]) = \frac{e^{-\Delta(p[i], p[i-1])^2 + O(p[i])^T W O(p[i-1])}}{\sum_j e^{-\Delta(s_j, p[i-1])^2 + O(s_j)^T W O(p[i-1])}}
\]
Experiments:

Yes.com playlists
• Dec 2010 – May 2011

“Small” dataset:
• 3,168 songs
• 134,431 + 1,191,279 transitions

“Large” dataset
• 9,775 songs
• 172,510 transitions + 1,602,079 transitions
Playlist prediction via Metric Embedding

Experiments:
Playlist prediction via Metric Embedding

Experiments:

Small

Big

Avg. log likelihood

\[ d \]

-14
-13
-12
-11
-10
-9
-8
-7
-6
-5

-14
-13
-12
-11
-10
-9
-8
-7
-6
-5

single-point LME

dual-point LME

Uniform

Unigram

Bigram
Morals of the story:

• Metric assumption works well in settings other than “geographical” data!
• However, they require some modifications in order to work well (e.g. “start points” and “end points”)
• Effective combination of latent + observed features, as well as metric + inner-product models
Efficient Natural Language Response Suggestion for Smart Reply

MATTHEW HENDERSON, RAMI AL-RFOU, BRIAN STROPE, YUN-HSUAN SUNG, LÁSZLÓ LUKÁCS, RUIQI GUO, SANJIV KUMAR, BALINT MIKLOS, and RAY KURZWEIL, Google

This paper presents a computationally efficient machine-learned method for natural language response suggestion. Feed-forward neural networks using n-gram embedding features encode messages into vectors which are optimized to give message-response pairs a high dot-product value. An optimized search finds response suggestions. The method is evaluated in a large-scale commercial e-mail application, Inbox by Gmail. Compared to a sequence-to-sequence approach, the new system achieves the same quality at a small fraction of the computational requirements and latency.

Additional Key Words and Phrases: Natural Language Understanding; Deep Learning; Semantics; Email

1 INTRODUCTION

Applications of natural language understanding (NLU) are becoming increasingly interesting with scalable machine learning, web-scale training datasets, and applications that enable fast and nuanced quality evaluations with large numbers of user interactions.

Early NLU systems parsed natural language with hand-crafted rules to explicit semantic representations, and used manually written state machines to generate specific responses from the output of parsing [18]. Such systems are generally limited to the situations imagined by the designer, and much of the development work involves writing more rules to improve the robustness of semantic parsing and the coverage of the state machines. These systems are brittle, and progress is slow [31].
Goal: Automatically suggest common responses to e-mails
Efficient Natural Language Response Suggestion for Smart Reply

Basic setup

new email $x$

Trigger suggestions? no

Response selection $(y_1, \ldots, y_k)$

Diversification $(y_{i1}, \ldots, y_{im})$

Response set $R$ and clustering

Smart Reply suggestions are shown

No Smart Reply suggestions
Efficient Natural Language Response
Suggestion for Smart Reply

Previous solution (KDD 2016)

- Based on a seq2seq method

\[
P(y | x) = P(y_1, \ldots, y_n | x_1, \ldots, x_m) = \prod_{i=1}^{n} P_{\text{LSTM}}(y_i | x_1, \ldots, x_m, y_1, \ldots, y_{i-1})
\]
Idea: Replace this (complex) solution with a simple multiclass classification-based solution

\[
P(y \mid x) = \frac{P(x, y)}{\sum_k P(x, y_k)}
\]

\[
P(x, y) \propto e^{S(x, y)}
\]

\[
P(\text{reply } y \mid \text{email } x) = \frac{e^{S(x, y)}}{\sum_y e^{S(x, y)}}
\]
Idea: Replace this (complex) solution with a simple multiclass classification-based solution.

\[ P_{\text{approx}}(y \mid x) = \frac{e^{S(x,y)}}{\sum_{k=1}^{K} e^{S(x,y_k)}} \]
Model: \( S(x,y) \)

\[ \Psi(x) \in \mathbb{R}^d \]

\[ \Psi(y) \in \mathbb{R}^d \]
Efficient Natural Language Response
Suggestion for Smart Reply

**Model:** Architecture v1

\[ S(x, y) = Wh \]

\[
\begin{align*}
\Psi(x) & \quad \Psi(y) \\
\text{ReLU layer} & \\
\text{ReLU layer} & \\
\text{ReLU layer} & \\
\text{ReLU layer} & \\
\end{align*}
\]

\[
\sigma \left( \sigma \left( \phi(x, y) \cdot \Lambda \right) \cdot \Lambda' \right) \theta
\]

\[
\phi(x) \quad \phi(y)
\]
Model: Architecture v2

\[ S(x, y) = h_x^T h_y \]
Efficient Natural Language Response Suggestion for Smart Reply

Model: Extensions

\[ S(x, y) = \mathbf{W}h \]

\[ S(x^i, y) = W^i h^i \]

\[ \forall i \]

multiple emails
**Model: Extensions**

<table>
<thead>
<tr>
<th>Message: Did you manage to print the document?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>With response bias</strong></td>
</tr>
<tr>
<td>-- Yes, I did.</td>
</tr>
<tr>
<td>-- Yes, it’s done.</td>
</tr>
<tr>
<td>-- No, I didn’t.</td>
</tr>
<tr>
<td><strong>Without response bias</strong></td>
</tr>
<tr>
<td>-- It’s printed.</td>
</tr>
<tr>
<td>-- I have printed it.</td>
</tr>
<tr>
<td>-- Yes, all done.</td>
</tr>
</tbody>
</table>

Diversity

More popular

Highly compatible, unpopular
Efficient Natural Language Response
Suggestion for Smart Reply

**Experiments:** (offline)

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Scoring Model</th>
<th>P@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>Joint</td>
<td>49%</td>
</tr>
<tr>
<td>25</td>
<td>Dot-product</td>
<td>48%</td>
</tr>
<tr>
<td>50</td>
<td>Dot-product</td>
<td>52%</td>
</tr>
</tbody>
</table>
## Experiments: (online)

<table>
<thead>
<tr>
<th>System</th>
<th>Experiment</th>
<th>Conversion rate relative to Seq2Seq</th>
<th>Latency relative to Seq2Seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exhaustive search</td>
<td>(1) Use a joint scoring model to score all responses in $R$.</td>
<td></td>
<td>500%</td>
</tr>
<tr>
<td>Two pass</td>
<td>(2) Two passes: dot-product then joint scoring.</td>
<td>67%</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>(3) Include response bias.</td>
<td>88%</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>(4) Improve sampling of dataset, and use multi-loss structure.</td>
<td>104%</td>
<td>10%</td>
</tr>
<tr>
<td>Single pass</td>
<td>(5) Remove second pass.</td>
<td>104%</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>(6) Use hierarchical quantization for search.</td>
<td>104%</td>
<td>1%</td>
</tr>
</tbody>
</table>
Morals of the story:

- Even a seemingly complex problem like natural-language response generation can be cast as a multiclass classification problem!
- Even a simple bag-of-words model proved to be sufficient, no need to handle “grammar” etc.
- Also, no personalization (though to what extent would this be possible with the data available?)
Learning Visual Clothing Style with Heterogeneous Dyadic Co-occurrences

Andreas Veit*, Balazs Kovacs*, Sean Bell, Julian McAuley, Kavita Bala, Serge Belongie

1 Department of Computer Science, Cornell University 2 Cornell Tech
3 Department of Computer Science and Engineering, UC San Diego

Abstract

With the rapid proliferation of smart mobile devices, users now take millions of photos every day. These include large numbers of clothing and accessory images. We would like to answer questions like ‘What outfit goes well with this pair of shoes?’ To answer these types of questions, one has to go beyond learning visual similarity and learn a visual notion of compatibility across categories. In this paper, we propose a novel learning framework to help answer these types of questions. The main idea of this framework is to learn a feature transformation from images of items into a latent space that expresses compatibility. For the feature transformation, we use a Siamese Convolutional Neural Network (CNN) architecture, where training examples are pairs of items that are either compatible or incompatible. We model compatibility based on co-occurrence in large-scale user behavior data; in particular, co-purchase data.

(a) Similar style as predicted by our model: items on the left are compatible with items on the right.

(b) Dissimilar style as predicted by our model: items on the left are incompatible with items on the right.
Goal: Identify items that might be purchased together
Learning Visual Clothing Style with Heterogeneous Dyadic Co-occurrences
Learning Visual Clothing Style with Heterogeneous Dyadic Co-occurrences

browsed together (substitutable)
bought together (complementary)
Four types of relationship:
1) People who viewed X also viewed Y
2) People who viewed X eventually bought Y
3) People who bought X also bought Y
4) People bought X and Y together

Substitutes (1 and 2), and Complements (3 and 4)
Learning Visual Clothing Style with Heterogeneous Dyadic Co-occurrences

1) Data collection

Shoes

Tops

Hems

Item images

Links

Categories
2) Training data generation

For each item $i$:
- Select one co-purchased item $j$.
- Select one non-co-purchased item $j'$.

such that $c(i) \neq c(j')$, $c(j) = c(j')$. 
2) Training
(simpler models)

\[ d(i, j) = \|\phi(i) - \phi(j)\|_0 \]
2) Training
(simpler models)
Learning Visual Clothing Style with Heterogeneous Dyadic Co-occurrences

3) Siamese CNNs
Learning Visual Clothing Style with Heterogeneous Dyadic Co-occurrences

4) Recommendation
Learning Visual Clothing Style with Heterogeneous Dyadic Co-occurrences
Learning Visual Clothing Style with Heterogeneous Dyadic Co-occurrences

Demo

User Study
Morals of the story:

- State-of-the-art recommender systems (whether from academia or industry) are not so far from what we learned in class.
- All of them depended on some kind of maximum-likelihood expression, along with gradient ascent/descent!