Part IV: Recommendation with Deep Learning

Xin Zhao
batmanfly@qq.com
Renmin University of China

Outline

- Overview of recommender systems
  - Tasks
    - Rating prediction
    - Item recommendation
  - Basic models
    - MF
    - libFM
- Deep learning for information mapping
  - DSSM
  - Cross-domain DSSM
  - Deep-Music
  - Deep-CTR
  - Google-Rec
- Deep learning for sequence modeling
  - Token2vec
    - POI recommendation
    - Product recommendation
  - Recurrent Neural Networks
    - POI recommendation

Overview of recommender systems

- Tasks
  - Rating prediction
  - Item recommendation
- Basic models
  - MF
  - LibFM

Rating Prediction

User-item matrix	i1	i2
u1	1	?
u2	?	5
u3	3	?
u4	?	2

- Online test
- Offline test

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

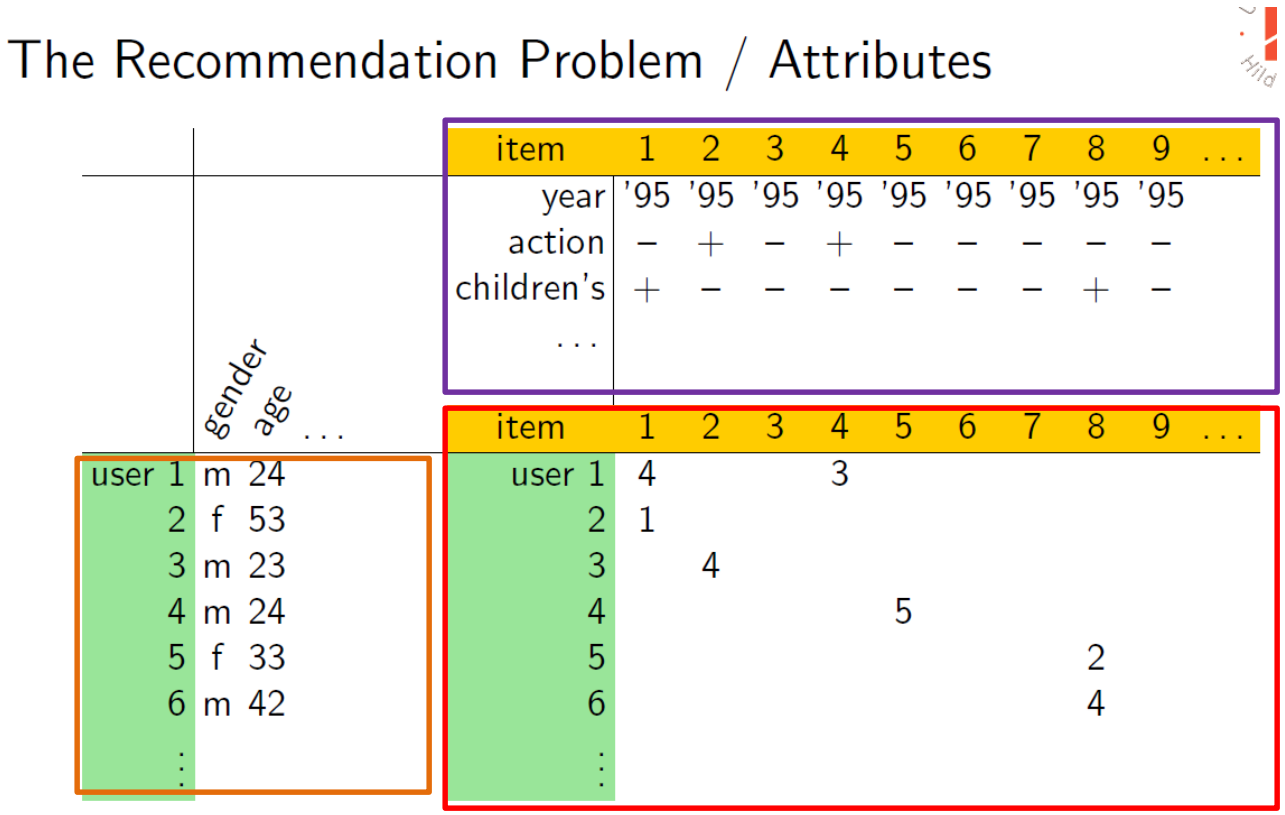
Item Recommendation

User-item matrix	i1	i2
u1	yes	?
u2	?	yes
u3	no	?
u4	yes	yes

- Online test
- Offline test
  - Retrieval-based metrics, e.g., P@k, R@k, MAP

Context-Aware Recommendation

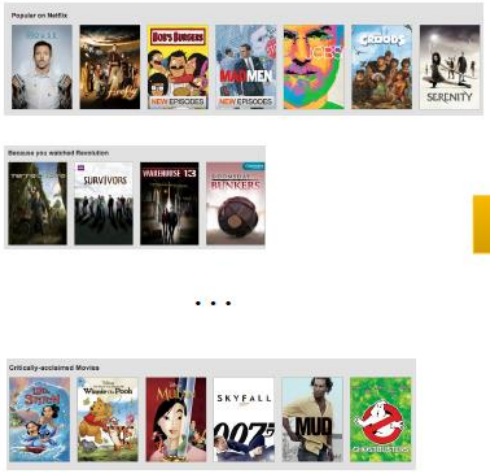
- When you know more information about users and items



More complicated tasks

Page Composition

10,000s of possible rows



Variable number of possible videos per row (up to thousands)

1 personalized page

10-40 rows

per device

Practical Considerations

- User Interface
- System requirements (efficiency, scalability, privacy....)
- Serendipity
- Diversity
- Awareness
- Explanations
- …

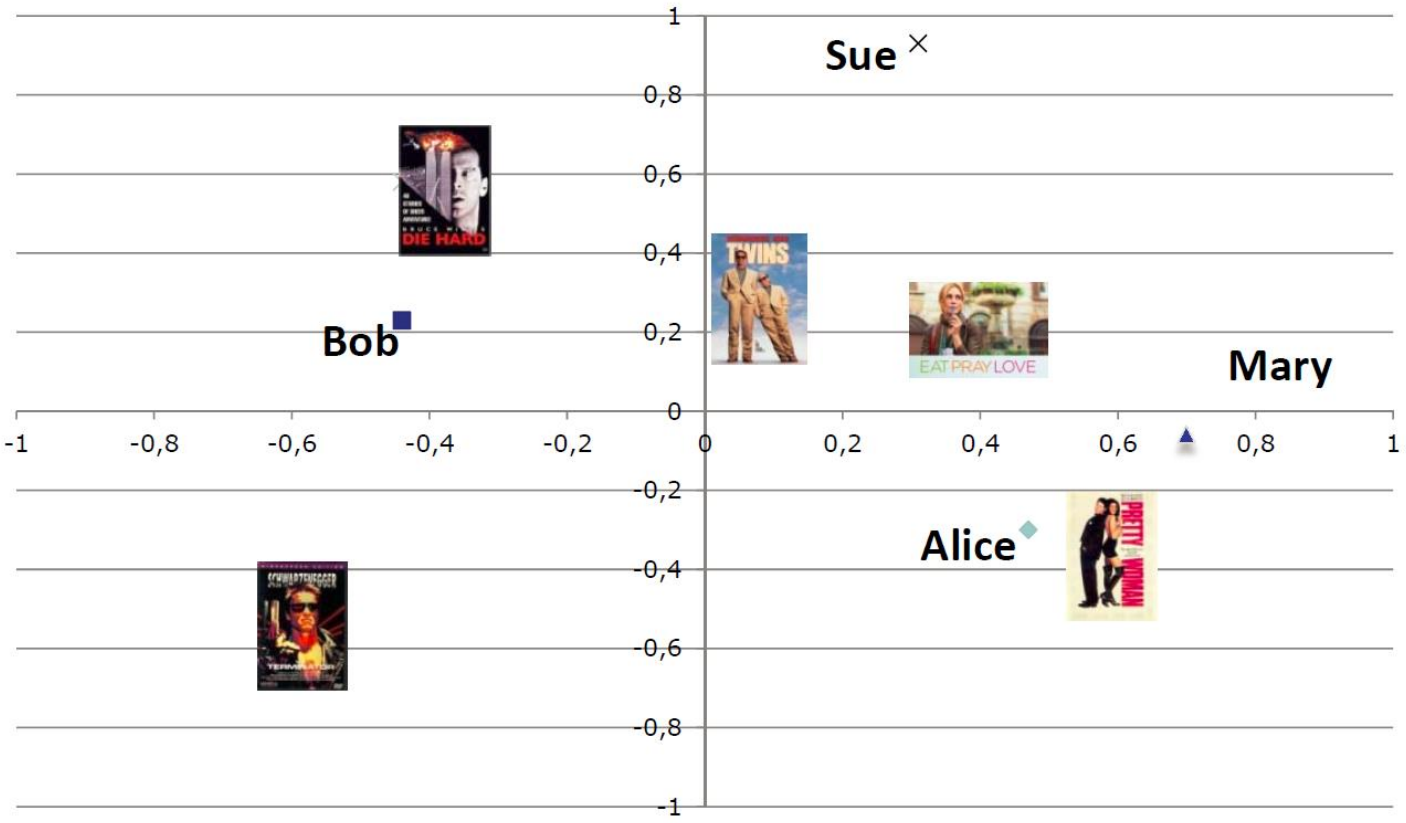
Rating Prediction

User-item matrix	i1	i2
u1	1	?
u2	?	5
u3	3	?
u4	?	2

- Online test
- Offline test

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Latent Factor Models



Matrix factorization

- **SVD:** $M_k = U_k \times \Sigma_k \times V_k^T$

U_k	Dim1	Dim2
Alice	0.47	-0.30
Bob	-0.44	0.23
Mary	0.70	-0.06
Sue	0.31	0.93

V_k^T					
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

Σ_k	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

- **Prediction:** $\hat{r}_{ui} = \bar{r}_u + U_k(Alice) \times \Sigma_k \times V_k^T(EPL)$

= 3 + 0.84 = 3.84

A Basic Model

$$\frac{1}{2}||R - U^{\mathsf{T}} V||_{\mathsf{F}}^2,$$

$$\min_{\mathsf{U},\mathsf{V}} \frac{1}{2} \sum_{\mathsf{i}=1}^{\mathsf{m}} \sum_{\mathsf{j}=1}^{\mathsf{n}} I_{\mathsf{ij}} \left(R_{\mathsf{ij}} - U_{\mathsf{i}}^{\mathsf{T}} V_{\mathsf{j}}\right)^2,$$

$$\min_{\mathsf{U},\mathsf{V}} \frac{1}{2} \sum_{\mathsf{i}=1}^{\mathsf{m}} \sum_{\mathsf{j}=1}^{\mathsf{n}} I_{\mathsf{ij}} \left(R_{\mathsf{ij}} - U_{\mathsf{i}}^{\mathsf{T}} V_{\mathsf{j}}\right)^2 + \frac{\lambda_1}{2}||U||_{\mathsf{F}}^2 + \frac{\lambda_2}{2}||V||_{\mathsf{F}}^2,$$

$$\min_{U,V} \| I \bigodot (R - U^{\top} V) \|_F^2 + \frac{\lambda_1}{2} \| U \|_F^2 + \frac{\lambda_2}{2} \| V \|_F^2$$

A Basic Model

Another formulation

$$\min_{U,V} \sum_{i,j} (r_{ij} - u_i^T v_j)^2 + \lambda_u \|u_i\|^2 + \lambda_v \|v_j\|^2$$

Probabilistic Matrix Factorization

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|U_i^T V_j,\sigma^2)\right]^{I_{ij}}$$

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

$$P(U,V|R,\sigma^2,\sigma_U^2,\sigma_V^2) = P(R|U,V,\sigma^2)P(U|\sigma_U^2)P(V|\sigma_V^2)$$

$$f_{\mathbf{x}}(x_1,\ldots,x_k) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right),$$

Probabilistic Matrix Factorization

$$p(R|U,V,\sigma^2) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|U_i^T V_j,\sigma^2)\right]^{I_{ij}}$$

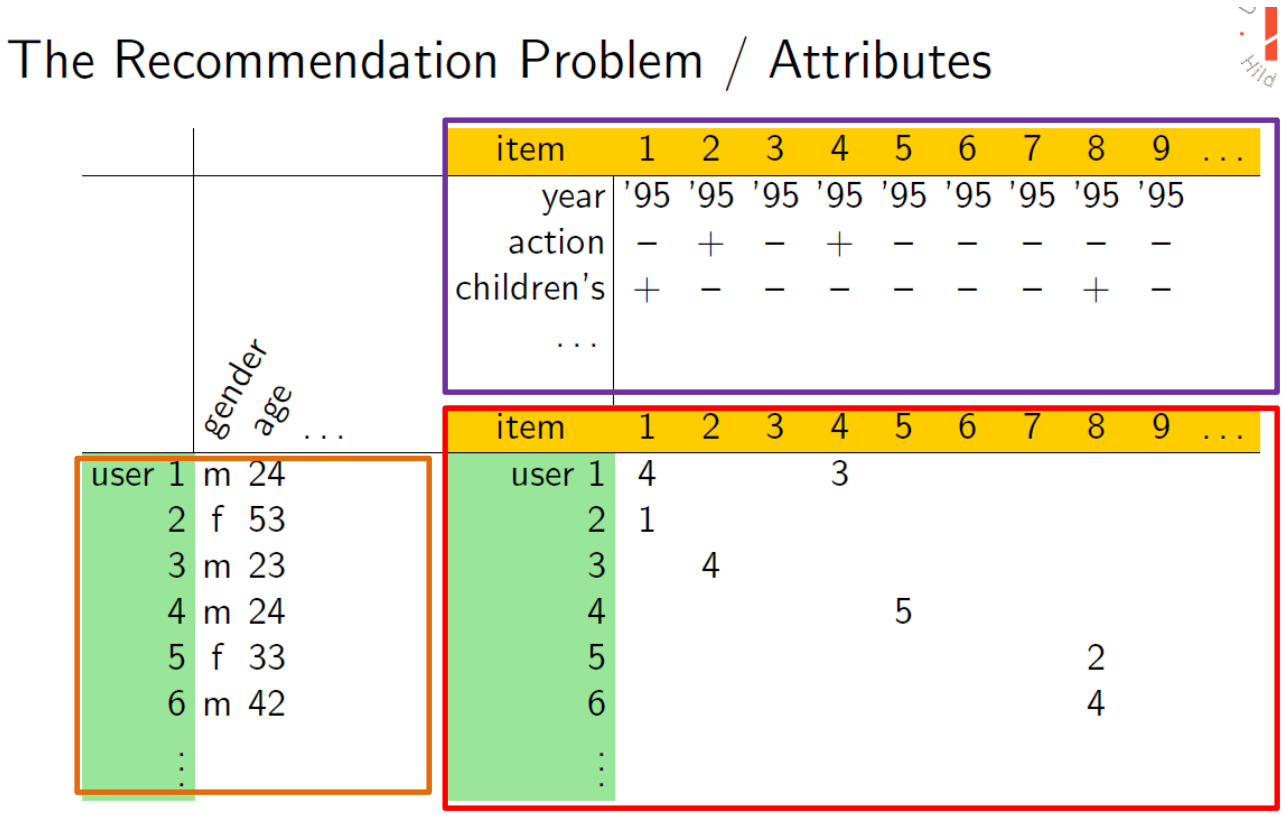
$$p(U|\sigma_U^2) = \prod_{i=1}^{N}\mathcal{N}(U_i|0,\sigma_U^2\mathbf{I}), \qquad p(V|\sigma_V^2) = \prod_{j=1}^{M}\mathcal{N}(V_j|0,\sigma_V^2\mathbf{I}).$$

$$\begin{aligned}\ln p(U,V|R,\sigma^2,\sigma_V^2,\sigma_U^2) = &-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\sum_{j=1}^{M}I_{ij}(R_{ij}-U_i^T V_j)^2 - \frac{1}{2\sigma_U^2}\sum_{i=1}^{N}U_i^T U_i - \frac{1}{2\sigma_V^2}\sum_{j=1}^{M}V_j^T V_j \\ &-\frac{1}{2}\left(\left(\sum_{i=1}^{N}\sum_{j=1}^{M}I_{ij}\right)\ln\sigma^2 + ND\ln\sigma_U^2 + MD\ln\sigma_V^2\right) + C, \qquad (3)\end{aligned}$$

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}I_{ij}\left(R_{ij}-U_i^T V_j\right)^2 + \frac{\lambda_U}{2}\sum_{i=1}^{N}\parallel U_i\parallel_{Fro}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M}\parallel V_j\parallel_{Fro}^2,$$

Context-Aware Recommendation

- When you know more information about users and items



LibFM

Factorization Machine (FM)

- Let $\mathbf{x} \in \mathbb{R}^p$ be an input vector with p predictor variables.
- Model equation (degree 2):

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$

- Model parameters:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^p, \quad \mathbf{V} \in \mathbb{R}^{p \times k}$$

Deep learning for information mapping

- DSSM
- Cross-domain DSSM
- Deep-Music
- Deep-CTR
- Google-Rec

Deep learning for information mapping

- **DSSM** (Huang et al., CIKM 2013)
- Cross-domain DSSM
- Deep-Music
- Deep-CTR
- Google-Rec

Deep Semantic Similarity Model

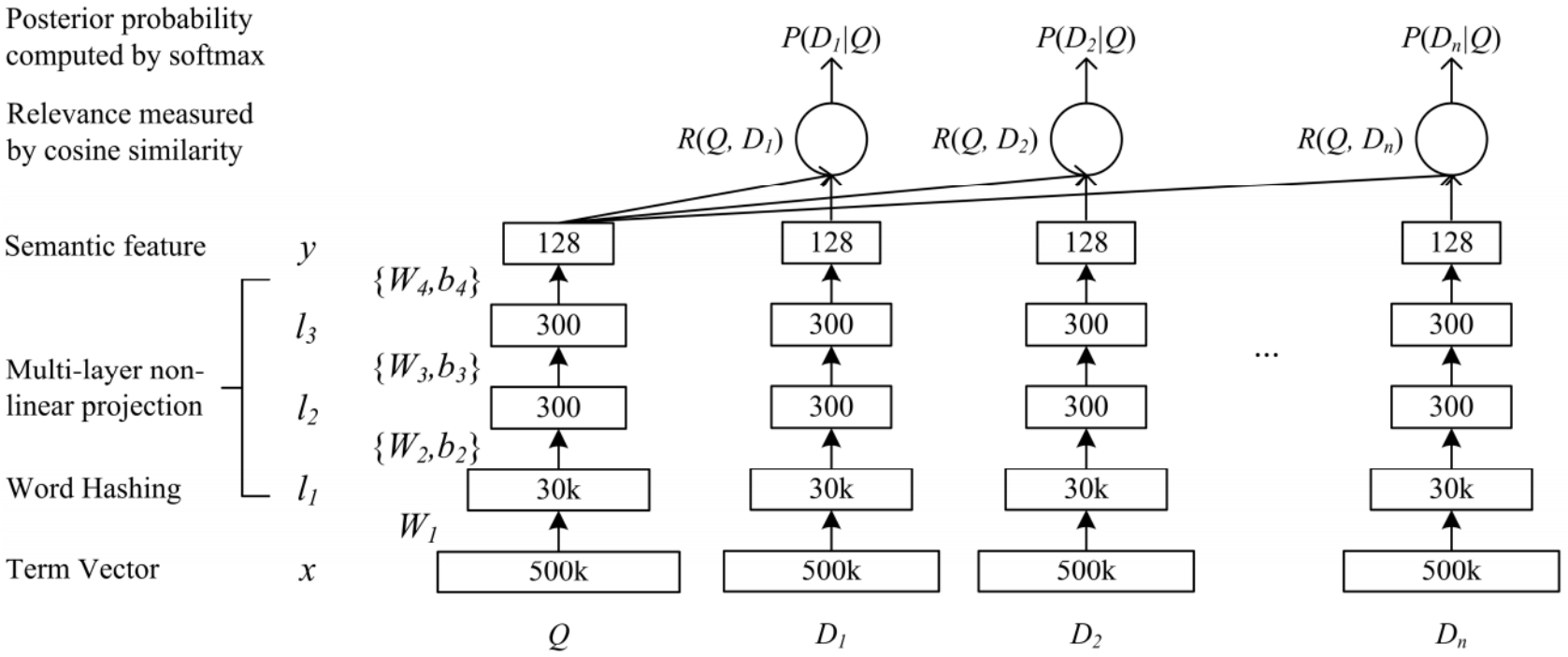
- Measuring similarity is the core problem in many tasks
  - We can borrow the similarity model from the text mining field for recommender systems
  - We start with text-based DSSM

Deep Semantic Similarity Model

- Information retrieval
  - Given a query ***Q***, we would like to return the most relevant documents ***D***
  - It usually adopts a ranking approach
  - A similarity function is needed for this purpose
    - $R(\boldsymbol{Q}, \boldsymbol{D})$

Deep Semantic Similarity Model

- DSSM for retrieval



Deep Semantic Similarity Model

- Reducing the number of features
  - Tri-letter hashing
  - For example, given a word #LOVE#, we will have
    - #LO, LOV, OVE, VE#

Deep Semantic Similarity Model

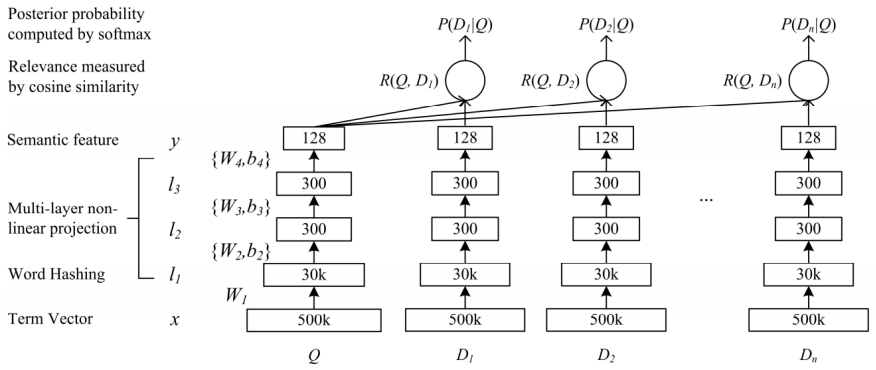
- Hidden space similarity measurement

The semantic relevance score between a query Q and a document D is then measured as:

$$R(Q,D) = \text{cosine}(y_Q, y_D) = \frac{{y_Q}^T y_D}{\|y_Q\| \|y_D\|} \quad (5)$$

$$P(D|Q) = \frac{\exp(\gamma R(Q,D))}{\sum_{D' \in \mathbf{D}} \exp(\gamma R(Q,D'))}$$

$$L(\Lambda) = -\log \prod_{(Q,D^+)} P(D^+|Q)$$



Deep Semantic Similarity Model

- Single-path modeling
  - A DNN component
  - From high-dimensional discrete feature vector to low-dimensional dense vector

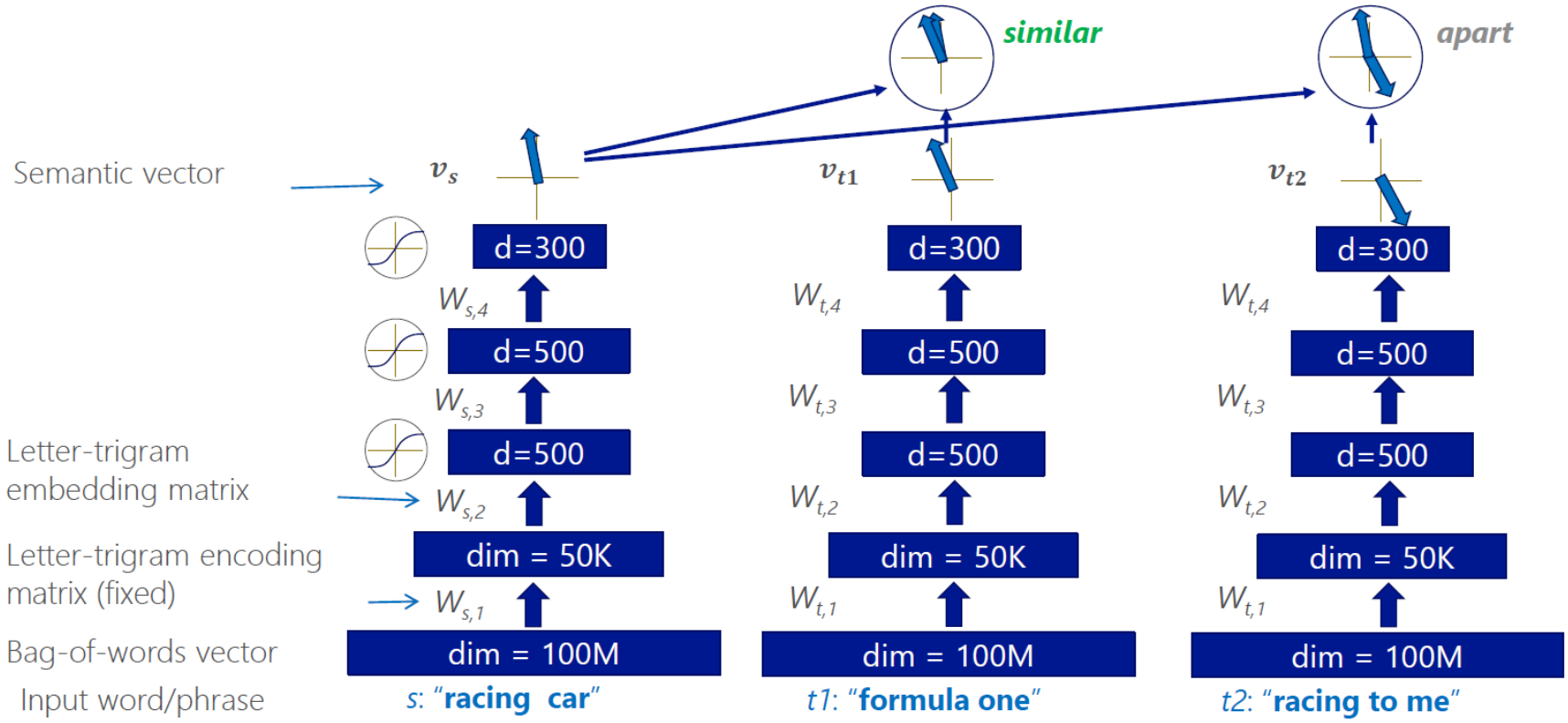
More formally, if we denote x as the input term vector, y as the output vector, l_i, $i = 1, \ldots, N-1$, as the intermediate hidden layers, W_i as the i-th weight matrix, and b_i as the i-th bias term, we have

$$
\begin{aligned}
l_1 &= W_1 x \\
l_i &= f(W_i l_{i-1} + b_i), i = 2, \ldots, N-1 \\
y &= f(W_N l_{N-1} + b_N)
\end{aligned}
\tag{3}
$$

where we use the $tanh$ as the activation function at the output layer and the hidden layers $l_i, i = 2, \ldots, N-1$:

$$
f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{4}
$$

Deep Semantic Similarity Model

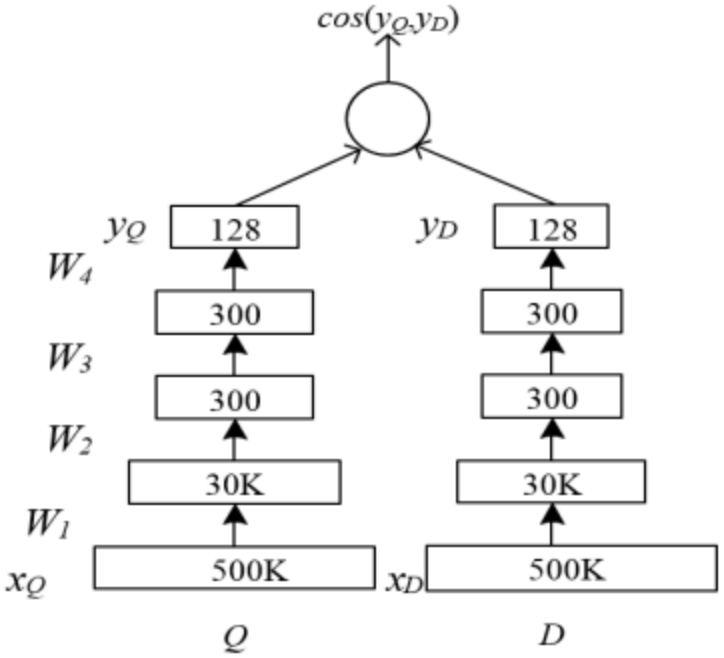


Deep Semantic Similarity Model

- Merits
  - Using DNN transformation, we can implement a powerful non-linear mapping function
  - Different NN components can be used and integrated
    - E.g., CNN

Single-domain DSSM Recommender

- DNN-based recommendation
  - Extracting features and building the NN architecture
  - If we have *N* domains, we build *N* DSSM separately



Deep learning for information mapping

- DSSM
- **Cross-domain DSSM** (Elkahky et al., WWW 2015)
- Deep-Music
- Deep-CTR
- Google-Rec

Cross-domain DSSM Recommender

- A user is usually with multiple types of behavior data

Type	DataSet	UserCnt	Feature Size	Joint Users
User View	Search	20M	3.5M	/
Item View	News	5M	100K	1.5M
	Apps	1M	50K	210K
	Movie/TV	60K	50K	16K

Table 1: Statistics of the four data sets used in this paper. The *Joint Users* column indicates the number of common users between each item view and the user view.

Cross-domain DSSM Recommender

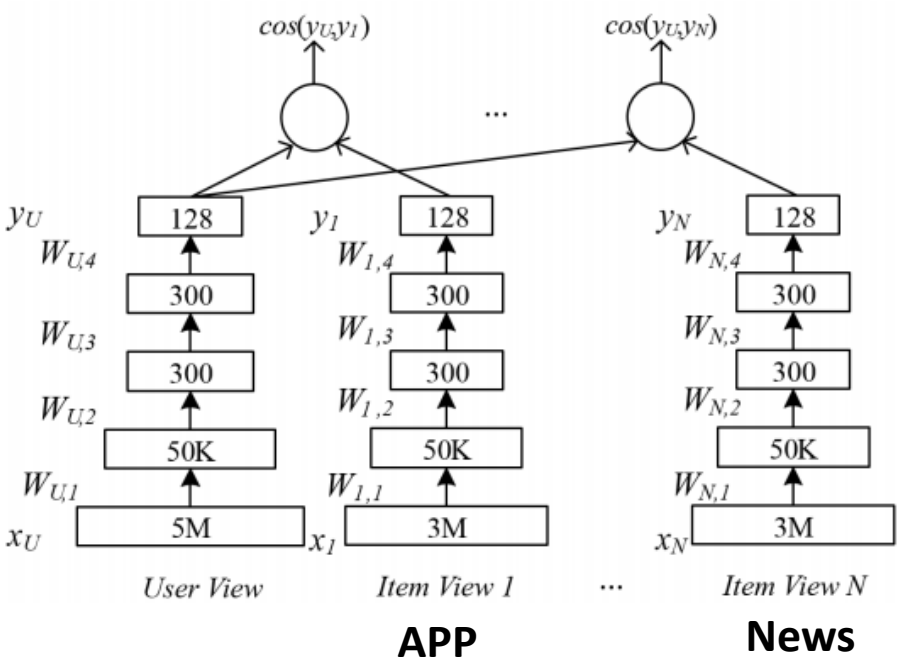
- A user is usually with multiple types of behavior data
  - Only using textual features
    - User features : users' search queries
    - News features: news titles
    - Apps: app titles
    - Movie/TV: title and descriptions

Type	DataSet	UserCnt	Feature Size	Joint Users
User View	Search	20M	3.5M	/
Item View	News	5M	100K	1.5M
	Apps	1M	50K	210K
	Movie/TV	60K	50K	16K

Table 1: Statistics of the four data sets used in this paper. The *Joint Users* column indicates the number of common users between each item view and the user view.

Cross-domain DSSM Recommender

- Cross-domain DSSM for recommendation
  - Joint recommending news, apps or movie/TV to users
  - User features are shared across domain



Deep learning for information mapping

- DSSM
- Cross-domain DSSM
- **Deep-Music** (Oord et al., NIPS 2013)
- Deep-CTR
- Google-Rec

Deep Music Recommender

- With training data
  - Representing users and songs in the same latent space
  - Predicting the final rating

$$\min_{U,V} \sum_{i,j} (r_{ij} - u_i^T v_j)^2 + \lambda_u \|u_i\|^2 + \lambda_v \|v_j\|^2$$

$\boldsymbol{u_i}$: user latent represnetation

$\boldsymbol{v_j}$: item latent represnetation

Deep Music Recommender

- Cold-start recommendation
  - How about new songs?
    - It cannot be learned with a latent factor

$$\min_{U,V} \sum_{i,j} (r_{ij} - u_i^T v_j)^2 + \lambda_u \|u_i\|^2 + \lambda_v \|v_j\|^2$$

For new songs, we do not have $\boldsymbol{v}_j$ at all.

Deep Music Recommender

- Cold-start recommendation
  - What we have for new songs?
    - Audio signals
    - Singer
    - Titles
    - Etc

$$\min_{U,V} \sum_{i,j} (r_{ij} - u_i^T v_j)^2 + \lambda_u \|u_i\|^2 + \lambda_v \|v_j\|^2$$

Coded as a song feature vector $\boldsymbol{a}_j$

Deep Music Recommender

- Cold-start recommendation
  - The idea is
    - For old songs, we have both audio features and the latent factors
    - Can we learn a mapping function from audio features to latent factors using old songs?

$$\min_{U,V} \sum_{i,j} (r_{ij} - u_i^T v_j)^2 + \lambda_u \|u_i\|^2 + \lambda_v \|v_j\|^2$$

$\boldsymbol{f}(\boldsymbol{a}_j) \rightarrow \boldsymbol{v}_j$

Coded as a song feature vector $\boldsymbol{a}_j$

Deep Music Recommender

- Cold-start recommendation
  - How to build such a mapping function $\boldsymbol{f(a_j) \rightarrow v_j}$
    - Yes, deep neural networks

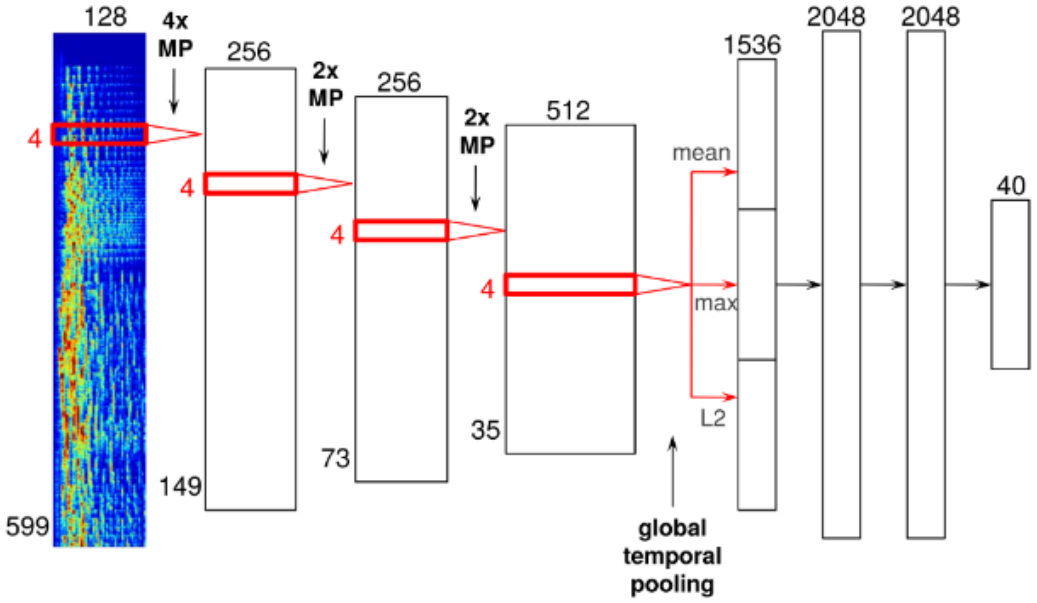
$$\min_{U,V} \sum_{i,j} (r_{ij} - u_i^T v_j)^2 + \lambda_u \|u_i\|^2 + \lambda_v \|v_j\|^2$$

$\boldsymbol{f(a_j) \rightarrow v_j}$

Coded as a song feature vector $\boldsymbol{a_j}$

Deep Music Recommender

- Cold-start recommendation
  - How to build such a mapping function $f(a_j) \rightarrow v_j$
    - Yes, deep neural networks



Deep learning for information mapping

- DSSM
- Cross-domain DSSM
- Deep-Music
- **Deep-CTR**
  - **CTR** (Wang et al., KDD 2013)
  - **Deep-CTR** (wang et al., KDD 2015)
- Google-Rec

Non-Deep Collaborative Topic Regression

- Recommendation setting
  - Recommending articles to users

$$\min_{U,V} \sum_{i,j} (r_{ij} - u_i^T v_j)^2 + \lambda_u \|u_i\|^2 + \lambda_v \|v_j\|^2$$

We can use basic MF methods, but contents are ignored
It suffers from cold-start problem

Non-Deep Collaborative Topic Regression

- Recommendation setting
  - Recommending articles to users

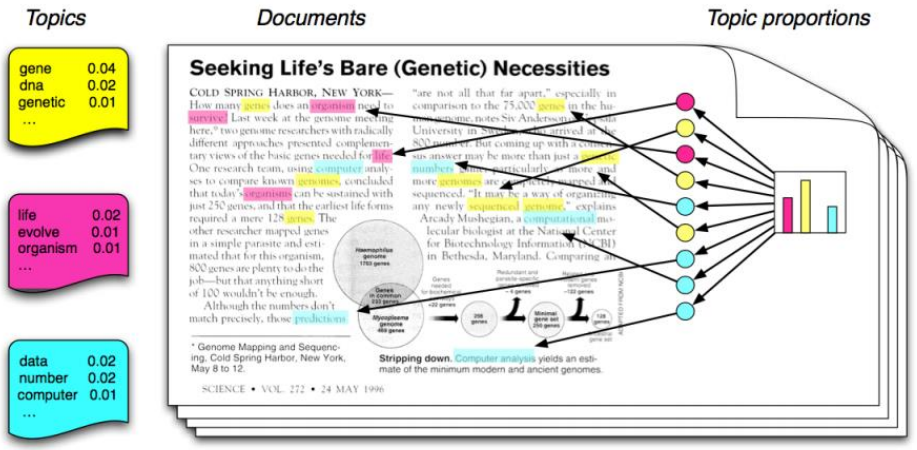
$$\min_{U,V} \sum_{i,j} (r_{ij} - u_i^T v_j)^2 + \lambda_u \|u_i\|^2 + \lambda_v \|v_j\|^2$$

We can use basic MF methods, but contents are ignored
It suffers from cold-start problem

If we would like to model contents, what model do we use?

Non-Deep Collaborative Topic Regression

- Candidate I: topic models



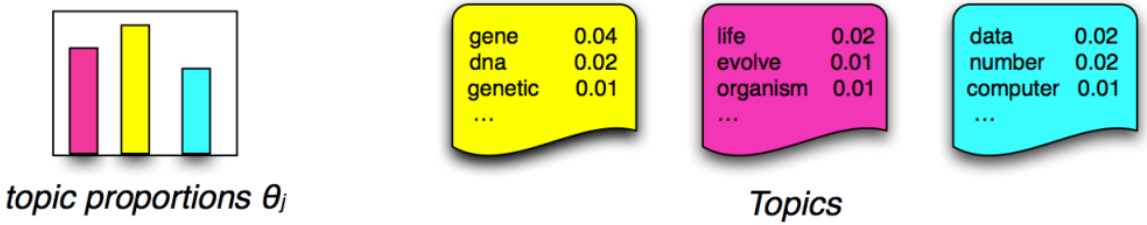
- Each topic is a distribution over words
- Each document is a mixture of topics
- Each word is drawn from one of those topics

Non-Deep Collaborative Topic Regression

- Candidate I: topic models

Latent Dirichlet allocation (LDA) is a popular topic model. It assumes

- There are K topics
- For each article, topic proportions $\theta \sim Dirichlet(\alpha)$



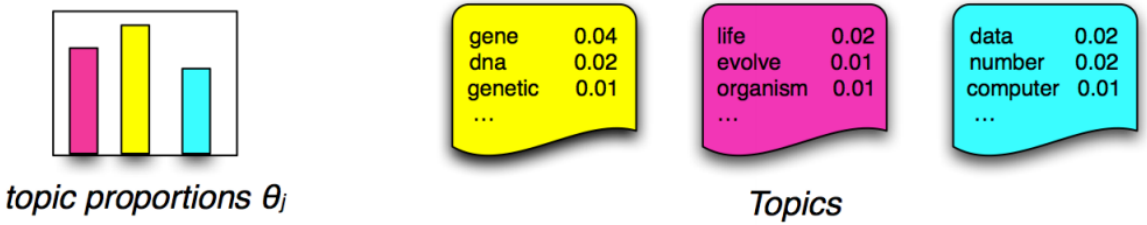
Note that θ can explain the topics that article talks about!

Non-Deep Collaborative Topic Regression

- Candidate I: topic models

Latent Dirichlet allocation (LDA) is a popular topic model. It assumes

- There are K topics
- For each article, topic proportions $\theta \sim Dirichlet(\alpha)$



Note that θ can explain the topics that article talks about!

Non-Deep Collaborative Topic Regression

- Candidate I: topic models

nodes	gene	distribution
wireless	genes	random
protocol	expression	probability
routing	tissues	distributions
protocols	regulation	sampling
node	coexpression	stochastic
sensor	tissuespecific	markov
peertopeer	expressed	density
scalable	tissue	estimation
hoc	regulatory	statistics

Maximum Likelihood from Incomplete Data via the *EM* Algorithm

By A. P. Dempster, N. M. Laird and D. B. Rubin

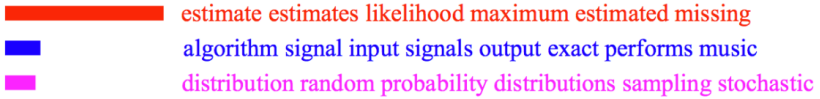
Harvard University and Educational Testing Service

[Read before the Royal Statistical Society at a meeting organized by the Research Section on Wednesday, December 8th, 1976, Professor S. D. Silvey in the Chair]

Summary

A broadly applicable algorithm for computing maximum likelihood estimates from incomplete data is presented at various levels of generality. Theory showing the monotone behaviour of the likelihood and convergence of the algorithm is derived. Many examples are sketched, including missing value situations, applications to grouped, censored or truncated data, finite mixture models, variance component estimation, hyperparameter estimation, iteratively reweighted least squares and factor analysis.

topic proportions



Non-Deep Collaborative Topic Regression

- Candidate I: topic models
  - We have two kinds of document representations

Maximum Likelihood from Incomplete Data via the *EM* Algorithm

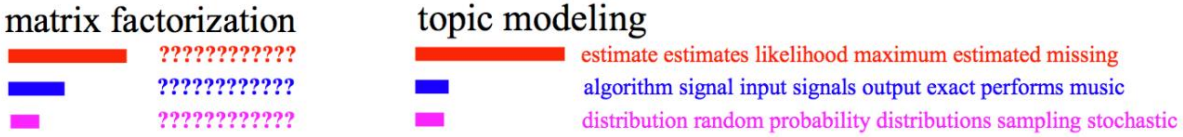
By A. P. Dempster, N. M. Laird and D. B. Rubin

Harvard University and Educational Testing Service

[Read before the Royal Statistical Society at a meeting organized by the Research Section on Wednesday, December 8th, 1976, Professor S. D. Silvey in the Chair]

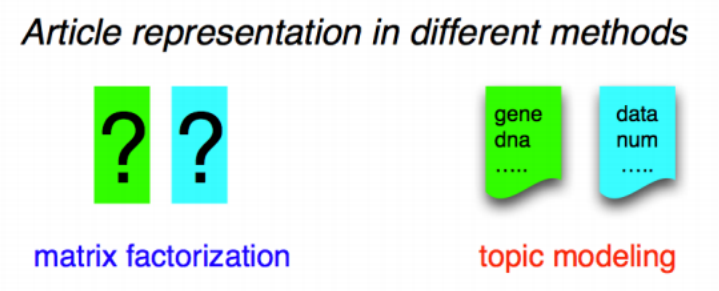
Summary

A broadly applicable algorithm for computing maximum likelihood estimates from incomplete data is presented at various levels of generality. Theory showing the monotone behaviour of the likelihood and convergence of the algorithm is derived. Many examples are sketched, including missing value situations, applications to grouped, censored or truncated data, finite mixture models, variance component estimation, hyperparameter estimation, iteratively reweighted least squares and factor analysis.



Non-Deep Collaborative Topic Regression

- Candidate I: topic models
  - The idea



- In matrix factorization, an article has a latent representation v in some *unknown latent space*
- In topic modeling, an article has topic proportions θ in the *learned* topic space

If we simply fix $v = \theta$, we seem to find a way to explain the unknown space using the topic space.

Non-Deep Collaborative Topic Regression

- Candidate I: topic models
  - The model

For each user i,

- Draw user latent vector $u_i \sim N(0, \lambda_u^{-1} I_k)$.

For each article j,

- Draw topic proportions $\theta_j \sim Dirichlet(\alpha)$.
- Draw item latent offset $\epsilon_j \sim N(0, \lambda_v^{-1} I_k)$ and set the item latent vector as $v_j = \theta_j + \epsilon_j$.
- Everything else is the same, the rating becomes,

$$E[r_{ij}] = u_i^T v_j = u_i^T (\theta_j + \epsilon_j).$$

This model is called Collaborative Topic Regression (CTR).

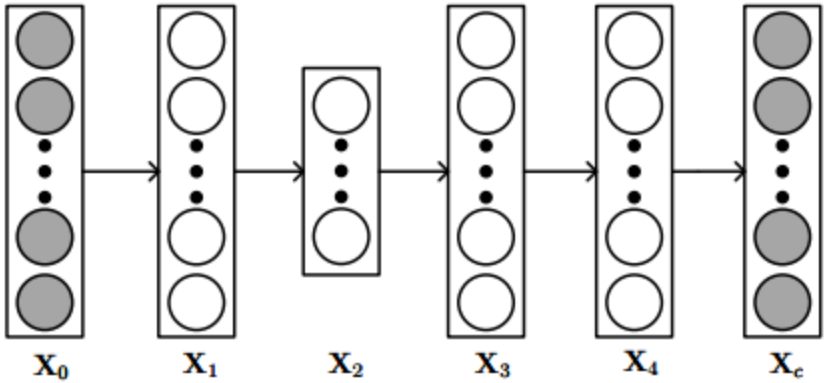
- Offset ϵ_j corrects θ_j for the popularity
- Precision parameter λ_v penalizes how much v_j could diverge from θ_j.

Deep Collaborative Topic Regression

- Candidate II: deep neural networks
  - We are essentially seeking an item representation based on its content
  - The content-based representation can be used or incorporated in recommendation

Deep Collaborative Topic Regression

- Candidate II: deep neural networks
  - Stacked Denoising Autoencoders



  - Inputting a term-based representation for articles, we can obtain the code for it

Deep Collaborative Topic Regression

- Candidate II: deep neural networks
  - The idea is nearly same to that of CTR

1. For each layer l of the SDAE network,
   (a) For each column n of the weight matrix $\mathbf{W}_l$, draw
   $$\mathbf{W}_{l,*n} \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l}).$$
   (b) Draw the bias vector $\mathbf{b}_l \sim \mathcal{N}(\mathbf{0}, \lambda_w^{-1}\mathbf{I}_{K_l})$.
   (c) For each row j of $\mathbf{X}_l$, draw
   $$\mathbf{X}_{l,j*} \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l), \lambda_s^{-1}\mathbf{I}_{K_l}).$$

2. For each item j,
   (a) Draw a clean input $\mathbf{X}_{c,j*} \sim \mathcal{N}(\mathbf{X}_{L,j*}, \lambda_n^{-1}\mathbf{I}_J)$.
   (b) Draw a latent item offset vector $\boldsymbol{\epsilon}_j \sim \mathcal{N}(\mathbf{0}, \lambda_v^{-1}\mathbf{I}_K)$ and then set the latent item vector to be:
   $$\mathbf{v}_j = \boldsymbol{\epsilon}_j + \mathbf{X}_{\frac{L}{2},j*}^T.$$
3. Draw a latent user vector for each user i:
   $$\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, \lambda_u^{-1}\mathbf{I}_K).$$
4. Draw a rating $\mathbf{R}_{ij}$ for each user-item pair (i, j):
   $$\mathbf{R}_{ij} \sim \mathcal{N}(\mathbf{u}_i^T\mathbf{v}_j, \mathbf{C}_{ij}^{-1}).$$

Deep Collaborative Topic Regression

- Candidate II: deep neural networks
  - Experiments

Table 1: mAP for three datasets

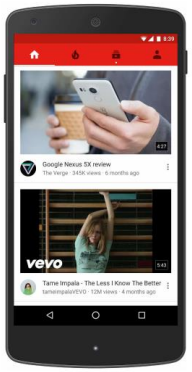
	citeulike-a	*citeulike-t*	*Netflix*
CDL	**0.0514**	**0.0453**	**0.0312**
CTR	0.0236	0.0175	0.0223
DeepMusic	0.0159	0.0118	0.0167
CMF	0.0164	0.0104	0.0158
SVDFeature	0.0152	0.0103	0.0187

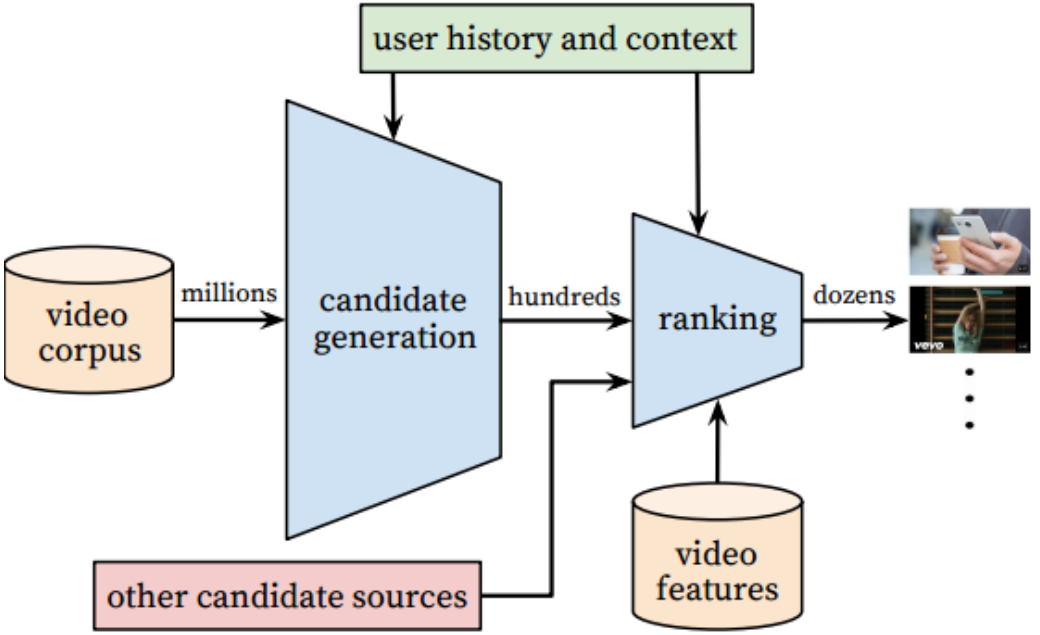
Deep learning for information mapping

- DSSM
- Cross-domain DSSM
- Deep-Music
- Deep-CTR
- **Google-Rec** (Covington et al., RecSys 2016)

Google's YouTube Recommendation

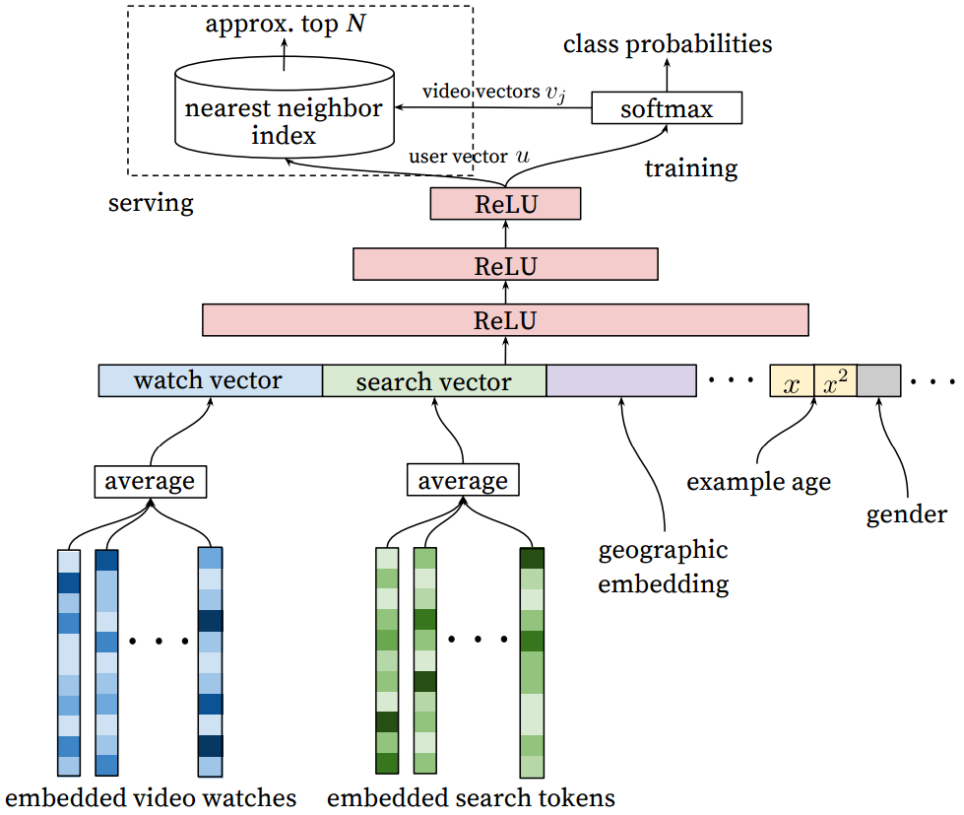
- Recommendation setting





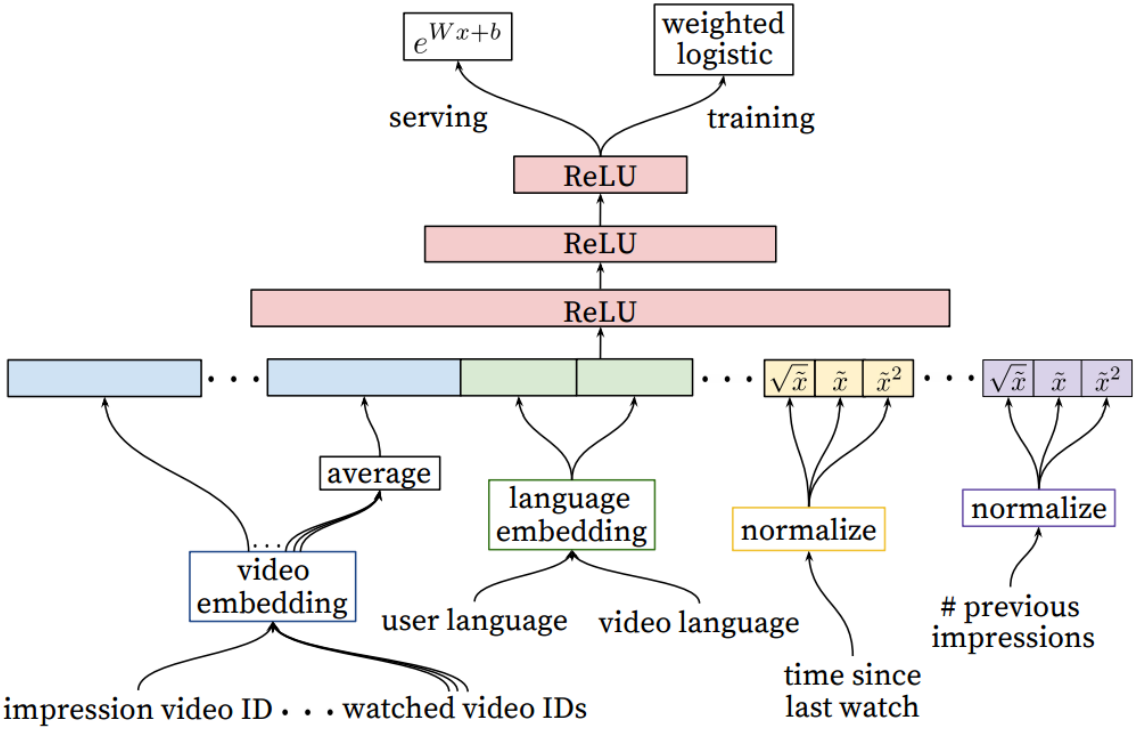
Google's YouTube Recommendation

- Deep candidate generation model



Google's YouTube Recommendation

- Deep ranking model



Deep learning for information mapping

- DSSM
  - A general semantic match model
- Cross-domain DSSM
  - Extension of DSSM to multiple-domain data
- Deep-Music
  - A deep content based model
- Deep-CTR
  - A deep content based model
- Google-Rec
  - Classification-based DNN model

Deep learning for sequence modeling

- Deep learning for sequence modeling
  - Token2vec
    - POI recommendation
    - Product recommendation
  - Recurrent Neural Networks
    - POI recommendation

Word2Vec

- Input: a sequence of words from a vocabulary ***V***
- Output: a fixed-length vector for each term in the vocabulary
  - $\mathbf{v}_w$

It implements the idea of distributional semantics using a shallow neural network model.

Token2Vec

- Input: a sequence of **symbol tokens** from a vocabulary ***V***

- Output: a fixed-length vector for each **symbol** in the vocabulary
  - $\mathbf{v}_w$

You can imagine that all the sequences in which surrounding contexts are sensitive can potentially be modeled with word2vec.

Check-in data

What information these check-in data contain?

User ID
Location ID
Check-in time
Category label/name
GPS information

Check-in data

What information these check-in data contain?

User ID
Location ID
Check-in time
Category label/name
GPS information

An example

UID25821
Burger King@BH Point
2015-01-13/1:30pm
Restaurant

A Sequential Way to Model the Data

- Given a user *u*, a trajectory is a sequence of check-in records related to *u*

User ID	Location ID	Check-in Timestamp
u1	l181	2016-08-26 9:26am
u1	l32	2016-08-26 10:26am
u1	l323	2016-08-26 11:26am
u1	l32323	2016-08-26 1:26pm
u2	l345	2016-08-26 9:16am
u2	l13	2016-08-26 10:36am

A Sequential Way to Model the Data

- Given a user *u*, a trajectory is a sequence of check-in records related to *u*

User ID	Location ID	Check-in Timestamp
u1	l181	2016-08-26 9:26am
u1	l32	2016-08-26 10:26am
u1	l323	2016-08-26 11:26am
u1	l32323	2016-08-26 1:26pm
u2	l345	2016-08-26 9:16am
u2	l13	2016-08-26 10:36am

u1: l181→l32 →l323 →l32323
u2: l345→l13

Deep learning for sequence modeling

- Deep learning for sequence modeling
  - Token2vec
    - **POI recommendation** (Zhao et al., TKDE 2016)
    - Product recommendation
  - Recurrent Neural Networks
    - POI recommendation

Task

- Input: Check-in sequences
- Output: Embedding representations for users, locations and other related information

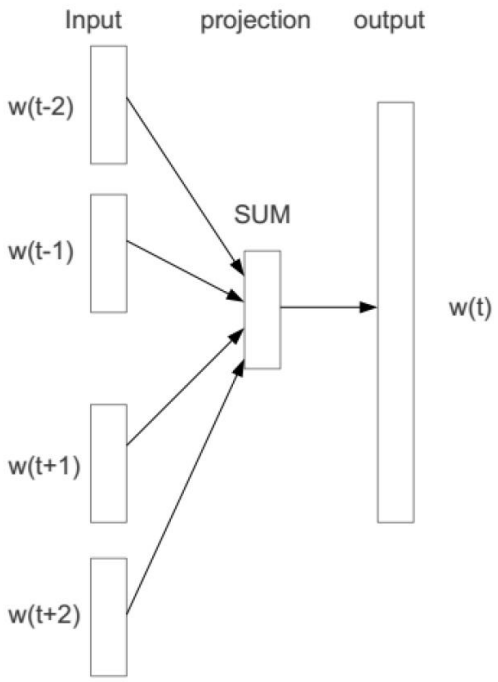
Motivations

- Trajectories are complicated
  - Much contextual information needs to be considered

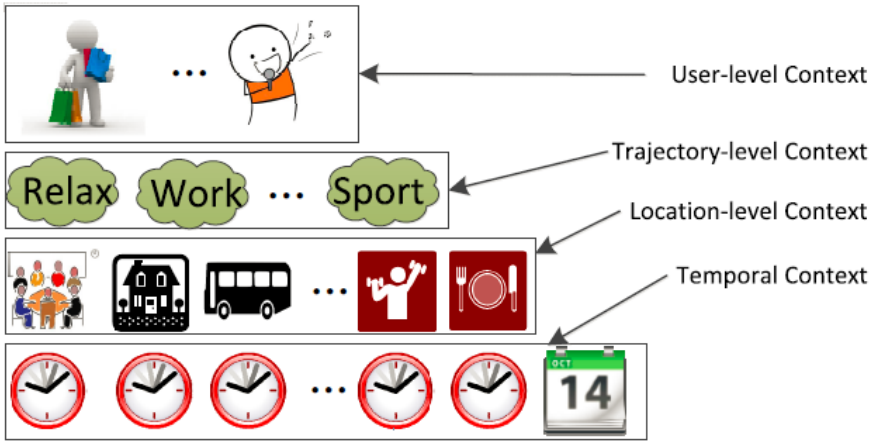
- Can we borrow the ideas in the work of **word embedding**?
  - A relatively cheap way to combine contexts
  - A simple, efficient yet effective approach

Recalling CBOW

- **CBOW** predicts the current word using surrounding contexts
  - $Pr(w_t \mid \text{context}(w_t))$
    - Window size 2c
    - $\text{context}(w_t) = [w_{t-c}, \ldots, w_{t+c}]$



Generation of a Single Location in a Trajectory



- User interests: u
- Trajectory intents: t
- Surrounding locations $\ell_{j-K} : \ell_{j+K}, c_{j-K} : c_{j+K}$
- Temporal contexts: d, h

Model

- A CBOW based framework

$$\frac{1}{N}\sum_{j=1}^{N}\log Pr(\ell_j|\boldsymbol{x}^{(\ell_j)}),$$

$\boldsymbol{x}^{(\ell_j)}$ is a real-valued feature vector consisting of all the contextual information for the *target location* ℓ_j

  - Coding the contextual information by using a feature vector
    - It is like the feature coding format in LibSVM

Model

- A *softmax* classifier

$$Pr(\ell_j|\boldsymbol{x}^{(\ell_j)}) = \frac{\exp(\bar{\boldsymbol{v}}_{\ell_j}^{\top} \cdot \boldsymbol{v}_{\ell_j})}{\sum_{\boldsymbol{v}'} \exp(\bar{\boldsymbol{v}}_{\ell_j}^{\top} \cdot \boldsymbol{v}')},$$

- Derive a context embedding vector

$$\bar{\boldsymbol{v}}_{\ell_j} = \frac{1}{\sum_f x_f^{(\ell_j)}} \sum_f x_f^{(\ell_j)} \times \boldsymbol{v}_f.$$

  - Each contextual feature corresponds to a unique embedding vector
  - Using simple weighted aggregation

The general model

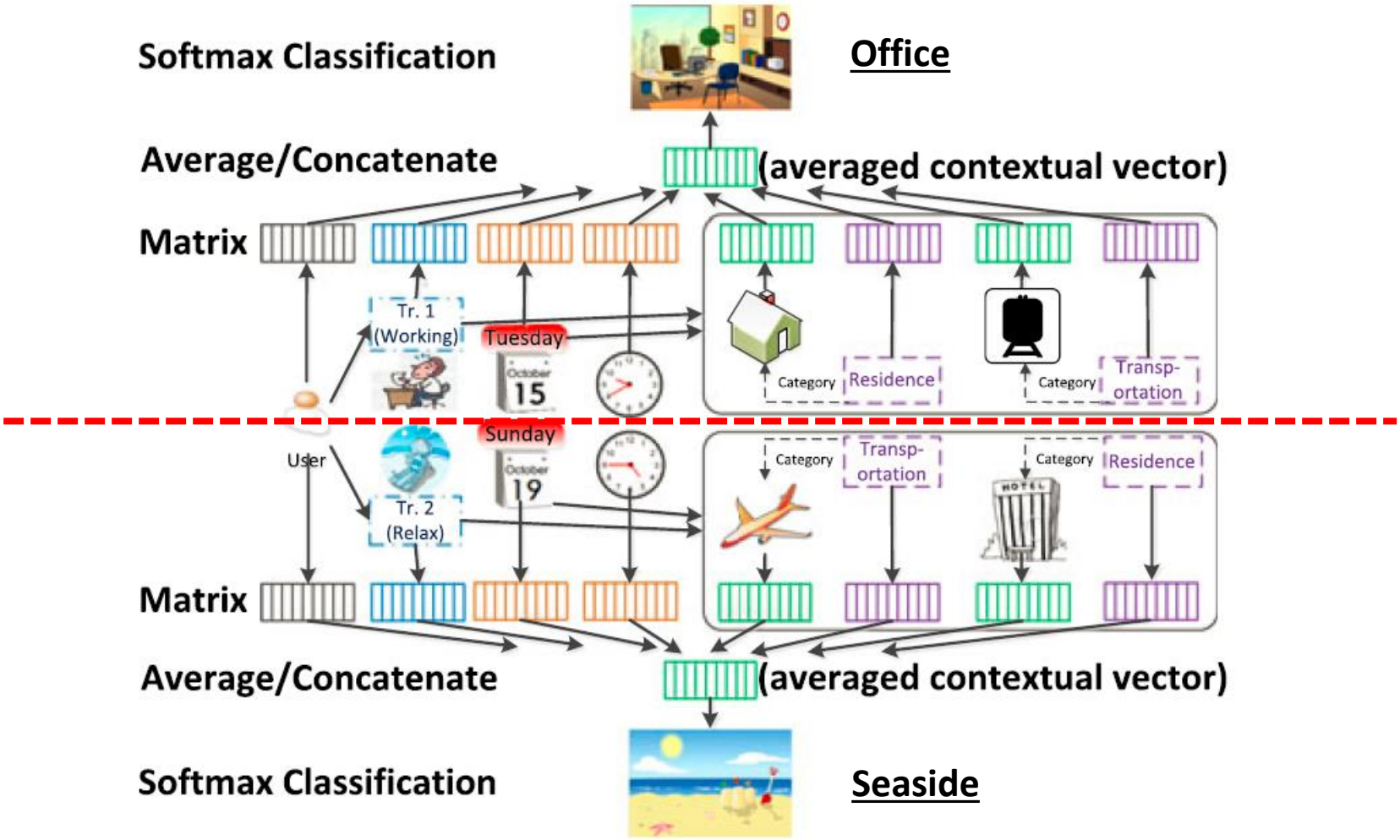
- Log-likelihood function

$$\sum_{u\in\mathcal{U}}\sum_{t\in\mathcal{T}^{(u)}}\frac{1}{N_t}\sum_{j=1}^{N_t}\log\Pr\Big(\ell_j|\underbrace{u}_{\text{user-level context}},\underbrace{t}_{\text{trajectory-level context}},\underbrace{\ell_{j-K}:\ell_{j+K},c_{j-K}:c_{j+K}}_{\text{location-level context}},\underbrace{d,h}_{\text{temporal context}}\Big),$$

  - General contextual embedding vector

$$\overline{\boldsymbol{v}}_{\ell_j}=\frac{1}{4K+4}\Big\{\sum_{-K\le k\le K,k\neq 0}\big(\boldsymbol{v}_{\ell_{j+k}}+\boldsymbol{v}_{c_{j+k}}\big)+\big(\boldsymbol{v}_u+\boldsymbol{v}_t+\boldsymbol{v}_d+\boldsymbol{v}_h\big)\Big\},$$

Illustrative example



Application I: Location recommendation

- General recommendation

$$S(u, \ell) \propto (\boldsymbol{v}_r + \boldsymbol{v}_u)^\top \cdot (\boldsymbol{v}_c + \boldsymbol{v}_\ell).$$

- Time-aware recommendation

$$S(u, \ell) \propto (\boldsymbol{v}_r + \boldsymbol{v}_{d_t} + \boldsymbol{v}_{h_j} + \boldsymbol{v}_u)^\top \cdot (\boldsymbol{v}_c + \boldsymbol{v}_\ell).$$

Application II: Social Link Prediction

- Given the trajectory data from users *u* and *v*, link prediction aims to predict whether there is a (reciprocal) link between *u* and *v*
  - A binary-classification approach
    - A pair of users
  - Represent a user using the embedding vector
    - Hadamard product between two user embedding vectors

$$\mathbf{x}^{u,v} = \mathbf{x}^u \circ \mathbf{x}^v$$
$$x_i^{u,v} = x_i^u \times x_i^v$$

Experiments

- Datasets

TABLE 2
Statistics of Our Datasets

Dataset	# Users	# Check-ins	# Links	# Locations
Foursquare$_S$	4,163	483,814	32,512	121,142
Foursquare$_L$	266,909	33,278,683	—	3,680,126
Gowalla	216,734	12,846,151	736,778	1,421,262

Location Recommendation

- General location recommendation

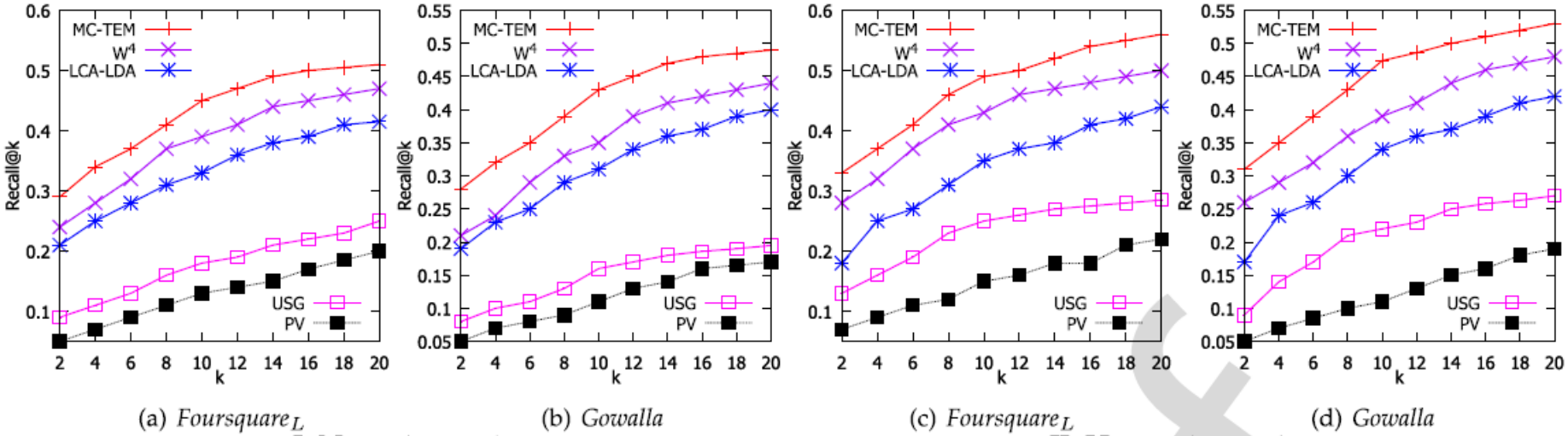
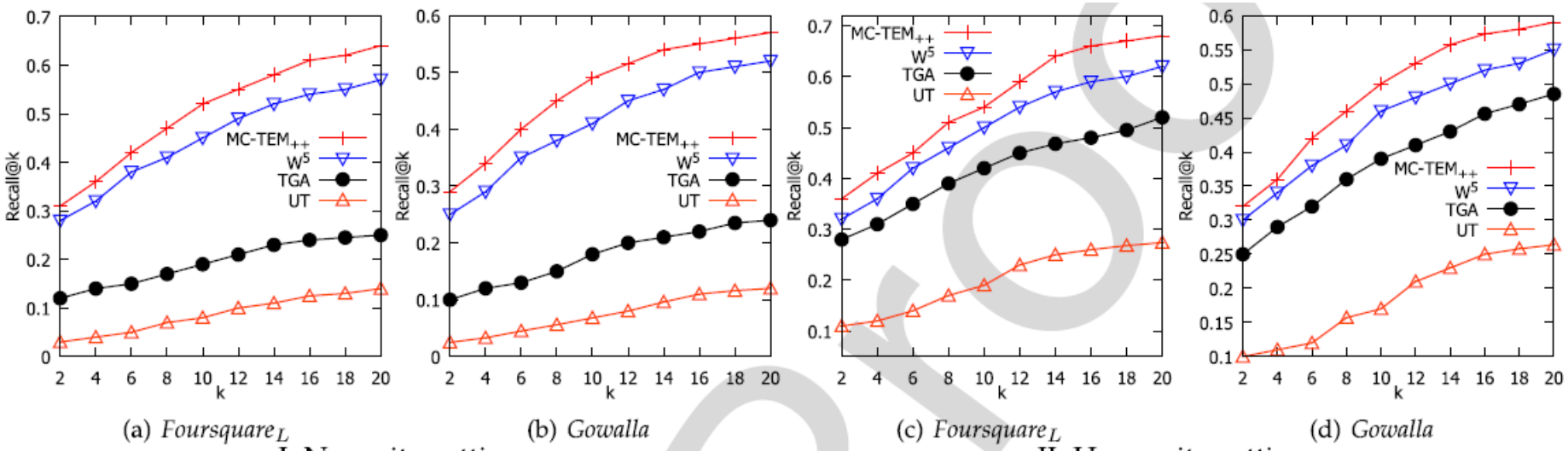


Fig. 5. Performance comparison on general location recommendation.

Location Recommendation

- Time-aware location recommendation



(a) *Foursquare*$_L$ (b) *Gowalla* (c) *Foursquare*$_L$ (d) *Gowalla*

I. New-city setting. II. Home-city setting.

Fig. 6. Performance comparison on time-aware location recommendation.

Social link prediction

TABLE 3
Performance Comparison on Link Prediction

Methods	*Foursquare*$_S$			*Gowalla*		
	P	R	F1	P	R	F1
MH	0.71	0.64	0.67	0.69	0.62	0.65
EBM	0.78	0.50	0.61	0.77	0.49	0.60
TSA	0.61	0.55	0.58	0.6	0.53	0.56
HMM	0.22	0.73	0.34	0.21	0.71	0.33
PV	0.56	0.31	0.40	0.54	0.29	0.37
MC-TEM	**0.81**	**0.75**	**0.77**	**0.79**	**0.74**	**0.76**

Qualitative examples

TABLE 5
Illustrative Examples for Query Context with the Corresponding Top Five Related Locations on the Foursquare$_S$ Dataset, Where ✓ Indicates the Location was Actually Visited by the User and ✘ Otherwise

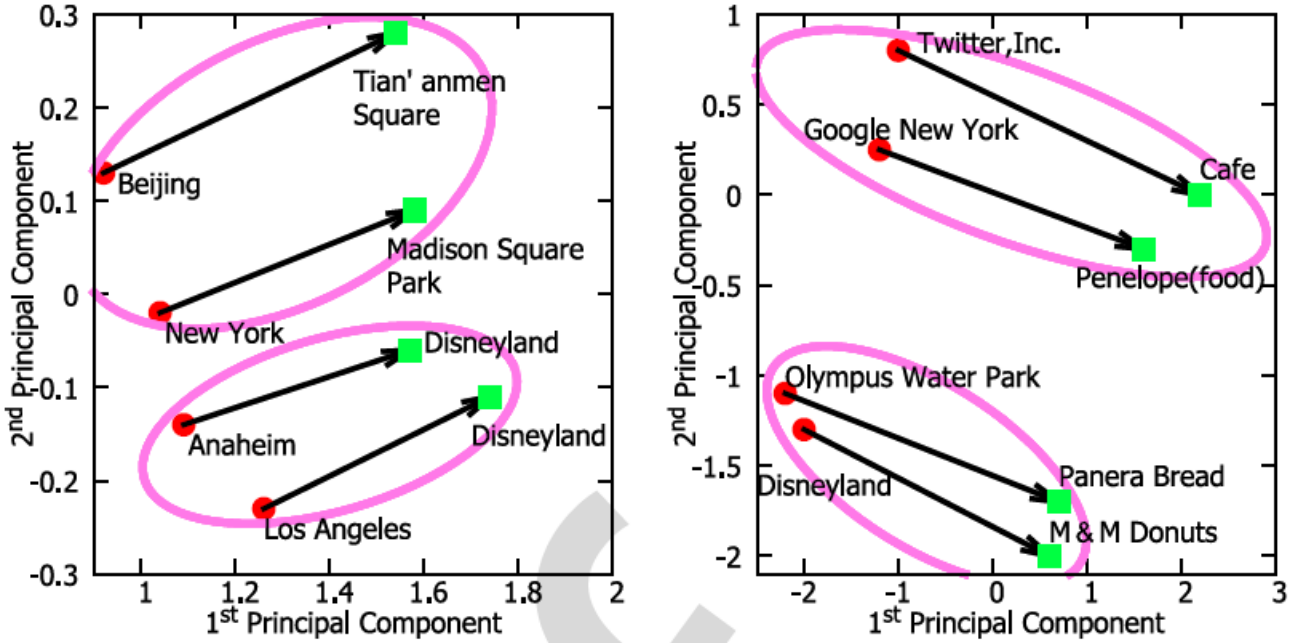
Type	Query Context	Top five related locations
Location	HK Star Seafood Resturant	Bo Ling's Chinese Restaurant$_{food}$, Soho Restaurant$_{food}$, Guadalajara Grill$_{food}$, Flavor Del Mar$_{food}$, Perch$_{food}$
Service Category	Food	Brown Owl Coffee$_{food}$, Teriyaki Maki$_{food}$, New Ca Mau Restaurant$_{food}$, Espresso Roma Cafe$_{food}$, Ajisen Ramen$_{food}$
	Museum	San Francisco Museum of Modern Art$_{museum}$, San Jose Museum of Art, MOCA$_{museum}$, The Metropolitan Museum of Art$_{museum}$, Griffith Observatory$_{museum}$
	7 o' clock	MTA Subway$_{travel}$, California Highway Patrol$_{travel}$, MUNI Bus Stop$_{travel}$, Metro Bus$_{travel}$, USA Gas Station$_{travel}$
Time	12 o'clock	Starbucks$_{food}$, McDonald's$_{food}$, Denny's$_{food}$, Subway$_{food}$, Burger King$_{food}$
	Tuesday	Twitter, Inc.$_{office}$, US Post Office$_{office}$, YouTube HQ$_{office}$, Bank of America$_{office}$, First Team SnS Real Estate$_{office}$, Chegg HQ$_{office}$
	Sunday	Times Square$_{recreation}$, Landmark Theatres$_{recreation}$, Runyon Canyon Park$_{recreation}$, Alcatraz Island$_{recreation}$, Macy's$_{recreation}$
City	Anaheim	Disneyland$_{recreation}$, Metrolink Anaheim Station$_{travel}$, Anaheim Hills Medical Center$_{office}$, Disney California Adventure Park$_{recreation}$, City National Grove of Anaheim$_{recreation}$
	San Diego	Balboa Park$_{recreation}$, San Diego International Airport$_{travel}$, San Diego Zoo$_{recreation}$, Port Of San Diego$_{travel}$, Fashion Valley Trolley Station$_{travel}$ and Transit Center$_{travel}$
User + Service Category	USER3584 + Food	KFC$_{food}$(✓), Vallejo's Restaurant$_{food}$(✓), Starbucks$_{food}$(✘), Roli Roti Gourmet Rotisserie$_{food}$(✓), Bistro Burger$_{food}$(✓)
	USER 3584 + Shop	Walgreens$_{shop}$(✓), Free People$_{shop}$(✓), Wells Fargo - Ocean$_{shop}$(✓), Old Navy$_{shop}$ (✘), Apple Store$_{shop}$(✓)

Observations in text data

- King – man = Queen – woman

- What about trajectory data?

Qualitative examples



(a) Type I: City *to* Location (b) Type II: Location *to* Location

Fig. 2. Identified examples for the pattern "A_1-$A_2 \approx B_1$-B_2" using the embedding vectors learnt on the Foursquare$_S$ [49] dataset. Each arrow line is associated with a pair of two entities. Each two corresponding entities pairs is in a ellipsoidal circle, indicating they have the approximately equal distance. For ease of visualization, PCA is used for dimensionality reduction, i.e., the first two principle components are used. See details in Section 5.4.

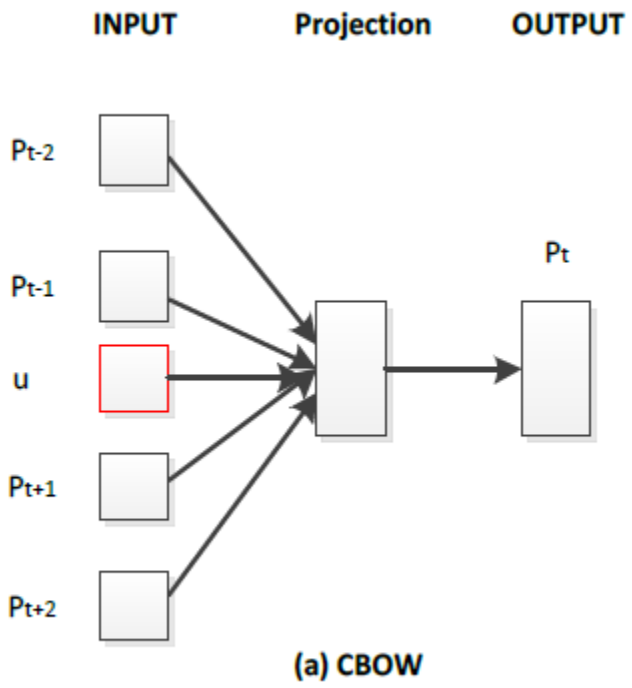
Deep learning for sequence modeling

- Deep learning for sequence modeling
  - Token2vec
    - POI recommendation
    - **Product recommendation**

      (Zhao et al., TKDE 2016, Wang et al., SIGIR 2015)
  - Recurrent Neural Networks
    - POI recommendation

Token2vec for Product Recommendation

- Doc2vec
  - Doc → user
  - Word → product

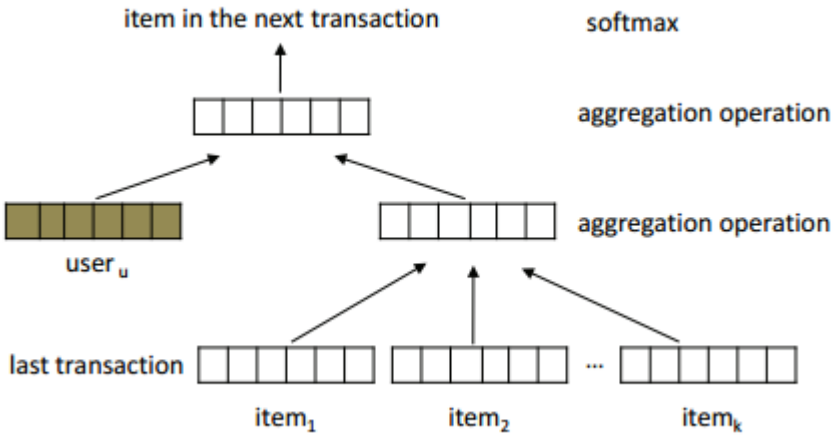


Token2vec for Product Recommendation

- Preliminary results on JingDong dataset
  - All the three simple embedding methods are comparative with the strong baseline BPR

	P@10	R@50	MAP	MRR
Doc2vec	0.152	0.527	0.330	0.546
LINE	**0.197**	**0.518**	**0.409**	**0.709**
DeepWalk	0.163	0.513	0.349	0.610

Token2vec for Product Recommendation



$$p(i \in T_t^u | u, T_{t-1}^u) = \frac{exp(\vec{v}_i^I \cdot \vec{v}_{u,t-1}^{Hybrid})}{\sum_{j=1}^{|I|} exp(\vec{v}_j^I \cdot \vec{v}_{u,t-1}^{Hybrid})}$$

where $\vec{v}_{u,t-1}^{Hybrid}$ denotes the hybrid representation obtained from the hierarchical aggregation which is defined as follows

$$\vec{v}_{u,t-1}^{Hybrid} := f_2(\vec{v}_u^U, f_1(\vec{v}_l^I \in T_{t-1}^u))$$

where $f_1(\cdot)$ and $f_2(\cdot)$ denote the aggregation operation at the first and second layer, respectively.

Neural Sequence Models

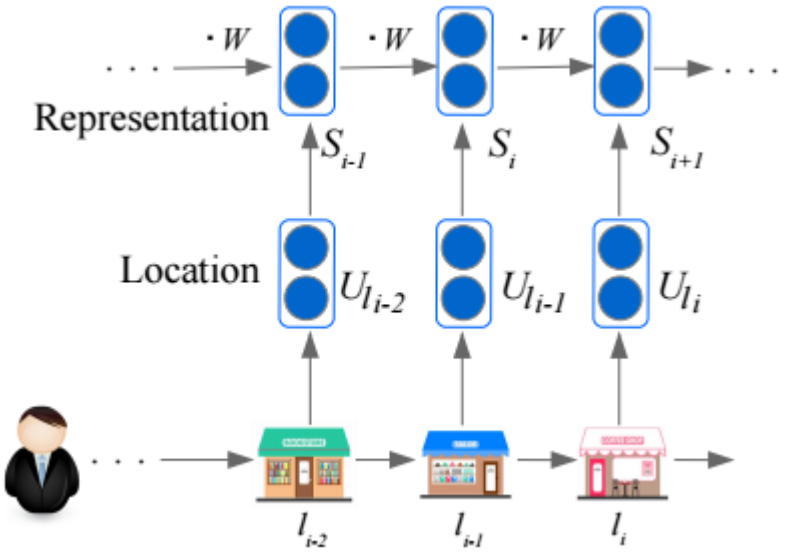
- Word2vec is a pseudo sequence model
- Only surrounding contexts are considered while the order of context words are ignored
- Recurrent neural network models can be used as an enhanced solution

Deep learning for sequence modeling

- Deep learning for sequence modeling
  - Token2vec
    - POI recommendation
    - Product recommendation
  - Recurrent Neural Networks
    - **POI recommendation** (Yang et al., arXiv 2016)

RNN for trajectory sequences

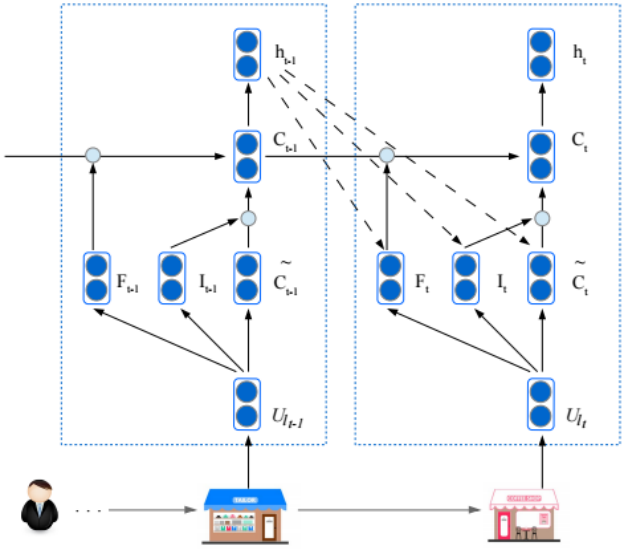
- In a short window



$$S_i = \tanh(U_{l_{i-1}} + W \cdot S_{i-1})$$

RNN for trajectory sequences

- In a long range, RNN tends to be less effective due to the problem of “vanishing gradient”
  - Long Short-Term Memory units (LSTM)
  - Gated Recurrent Unit (GRU)



$$\widetilde{C_t} = \tanh(W_{c_1} U_{l_t} + W_{c_2} h_{t-1} + b_c)$$

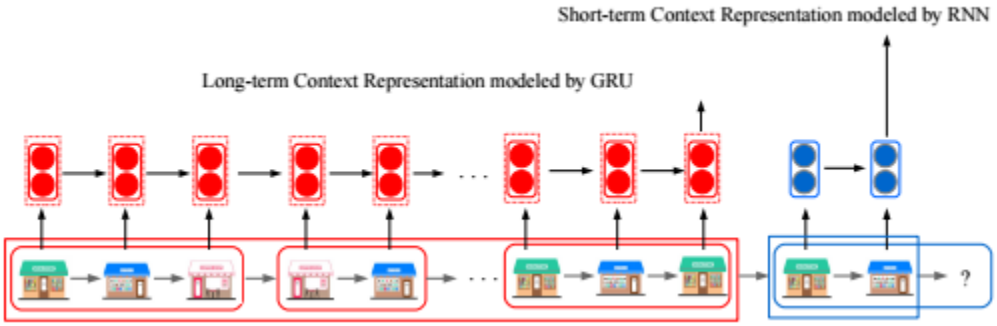
$$C_t = i_t * \widetilde{C_t} + f_t * C_{t-1}$$

$$i_t = \sigma(W_{i_1} U_{l_t} + W_{i_2} h_{t-1} + b_i)$$

$$f_t = \sigma(W_{f_1} U_{l_t} + W_{f_2} h_{t-1} + b_f)$$

RNN for trajectory sequences

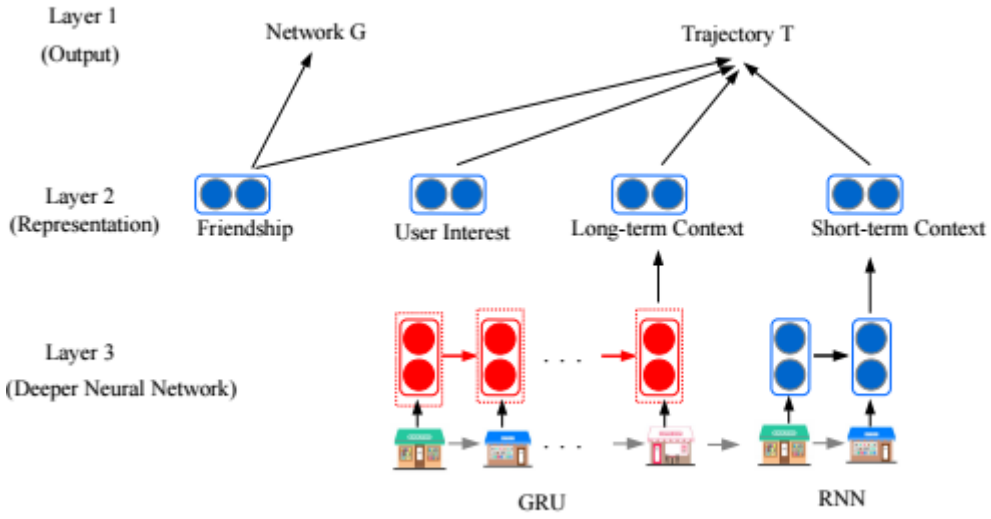
- Combine short- and long-term dependence together



$$\mathcal{L}(T_v) = \sum_{i=1}^{m} \log \Pr[l_i^{(v,j)} | \underbrace{l_1^{(v,j)} : l_{i-1}^{(v,j)}}_{\text{short-term contexts}}, \underbrace{T_v^1 : T_v^{j-1}}_{\text{long-term contexts}}, v, \Phi].$$

RNN for trajectory sequences

- Incorporate user interests and networks



Conclusions

- Data models are the core to different task in multiple fields. Try to transfer techniques around fields.
- Sequence models are very useful to characterize users' behaviors
- Shallow neural networks have good practical performance and quick running speed.

Thanks!

batmanfly@qq.com

References

- 深度学习在推荐算法中的研究进展.赵鑫. 中国人工智能学会通讯.2016年第七期.
- Yehuda Koren, Robert M. Bell, Chris Volinsky. Matrix Factorization Techniques for Recommender Systems. IEEE Computer 42(8): 30-37 (2009)
- Steffen Rendle. Factorization Machines with libFM. ACM TIST 3(3): 57 (2012)
- Xavier Amatriain, Bamshad Mobasher. The recommender problem revisited: tutorial. KDD 2014
- Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, Larry P. Heck. Learning deep structured semantic models for web search using clickthrough data. CIKM 2013: 2333-2338
- Ali Mamdouh Elkahky, Yang Song, Xiaodong He. A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems. WWW 2015: 278-288
- Aäron Van Den Oord, Sander Dieleman, Benjamin Schrauwen. Deep content-based music recommendation. NIPS 2013: 2643-2651
- Chong Wang, David M. Blei. Collaborative topic modeling for recommending scientific articles. KDD 2011: 448-456
- Hao Wang, Naiyan Wang, Dit-Yan Yeung. Collaborative Deep Learning for Recommender Systems. KDD 2015: 1235-1244
- Paul Covington, Jay Adams, Emre Sargin. Deep Neural Networks for YouTube Recommendations. RecSys 2016: 191-198
- Wayne Xin Zhao**, Ningnan Zhou**, Xiao Zhang, Ji-Rong Wen, Shan Wang. A General Multi-Context Embedding Model for Mining Human Trajectory Data. IEEE Trans. Knowl. Data Eng. 28(8): 1945-1958 (2016)
- Wayne Xin Zhao, Sui Li, Yulan He, Edward Y. Chang, Ji-Rong Wen, Xiaoming Li. Connecting Social Media to E-Commerce: Cold-Start Product Recommendation Using Microblogging Information. IEEE Trans. Knowl. Data Eng. 28(5): 1147-1159 (2016)
- Quoc V. Le, Tomas Mikolov. Distributed Representations of Sentences and Documents. ICML 2014: 1188-1196
- Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, Xueqi Cheng. Learning Hierarchical Representation Model for NextBasket Recommendation. SIGIR 2015: 403-412
- Cheng Yang, Maosong Sun, Wayne Xin Zhao, Zhiyuan Liu. A Neural Network Approach to Joint Modeling Social Networks and Mobile Trajectories. arXiv:1606.08154 (2016)

Disclaimer

- For convenience, I directly copy some original slides or figures from the referred papers. I am sorry that I did not ask for the permission of each referred author. I thank you for these slides. I will not distribute your slides.