Recommendation Systems for Online Advertising

SUMIT SIDANA

Supervised by:
MASSIH-REZA AMINI
Co-Supervised by:
CHARLOTTE LACLAU

08/11/2018

FUI Project Calypso

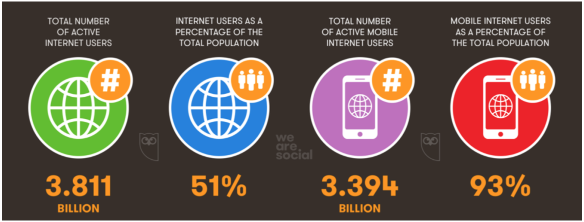
Goal Design of an efficient decision system for online advertising

Participants (not an exhaustive list!)

- **Kelkoo Group**: eCommerce marketing platform
  - Jerome Colin (Program manager)
  - Gilles Vandelle (Chief scientist)
  - André Brois-Crettez (Software Architect / Data Scientist)
- **Purch**: digital publishing and marketplace platform
  - Laurent Metzger (Director of Engineering)
  - Christophe Sebastien (Senior Software Developer)
  - Abderrazagh Mbodj (Data Scientist)
- **LIG**: Laboratory of computer science of Grenoble
  - Massih-Reza Amini (Professor)
  - Charlotte Laclau (Maître de Conférences)
  - Sumit Sidana (Ph.D. student)

Machine learning for online advertising/recommender systems: a huge market

Context: more and more people use internet



Strong economic impact for major companies

More than ...

- 60% of the films seen on **Netflix** are recommended ones
- 35% of sales on **Amazon** are done thanks to recommendation
- 38% of clicks on **Google** are generated over recommended items

The "Recommender problem"

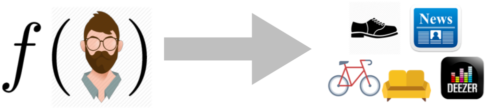
Traditional definition

- Estimate a utility function that automatically predicts how much a user will like an item.

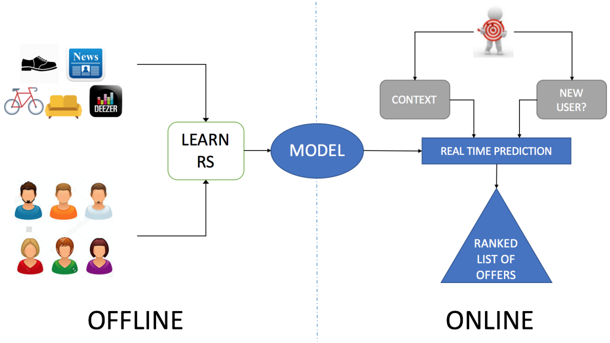
Based on:

- Past behavior
- Relation to other users
- Item similarity
- Context
- ...

What all items will this user like:

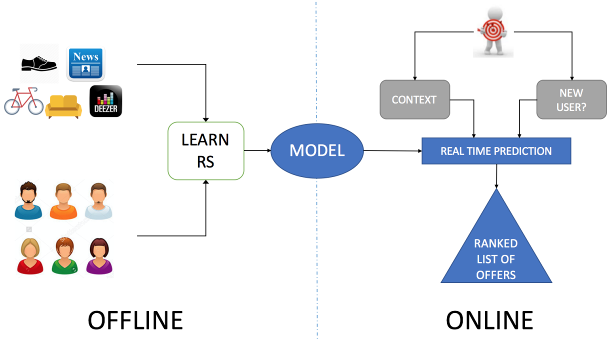


Recommendation: A two stage process



- **Stage 1**: gather data and learn a model (offline)
- **Stage 2**: use the model to make prediction (online)

Recommendation: A two stage process



- **Stage 1**: gather data and learn a model (offline)
- **Stage 2**: use the model to make prediction (online)

In this Ph.D., we focus mostly on offline stage

Collaborative filtering

CF: set of techniques ignoring user and item attributes but rather focusing on **user-item interactions.**

Memory-based

- Use the data to establish correlation between users or items
- Recommend from items seen by the similar users
- Do not work well under sparsity constraints or scale very well

Model-based

- Also known as *latent factor models*
- Attempts to build a model to explain the rating patterns

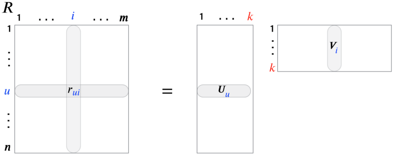


Figure: Principle of Matrix Factorization.

Learning to rank - Overview classification

→ Care to make accurate **ranking** and not **rating** prediction
→ Learning-to-Rank for recommendations is a more realistic problem to solve, as compared to, the rating prediction problem

Pointwise: train a **regressor** to make predictions then rank the results from higher to lower
Pairwise: look at **pair of items** and try to predict relative order for that pair
Listwise: look at the **entire list of items** and try to come up with the optimal ordering of it

Pointwise	Pairwise	Listwise
Matrix Factorization [Koren 2009]	**BPR** [Rendle 2009]	**CofiRank** [Weimer 2007]
Factorisation Machines [Rendle 2009]	**EigenRank** [Liu 2008]	**ListRank** [Shi 2010]
Field-aware FM [Juan 2016]	**LightFM** [Kula 2015]	**WLT** [Volkovs 2012]

Representation Learning (RL) with Embeddings

Embedding as applied to words

- Research in vector representations of words has taken off since the work of [Mikolov et al., 2013], who represent words as embedding vectors.

Representation Learning with Neural Networks

- Sparse datasets represented as low dimensional dense vectors
- Exploiting the sequential nature of user's interactions
  - **Prod2Vec** [Grobovic 2015]: next item prediction
  - **Neural Collaborative Filtering** [He et al., 2017]: A neural architecture combining GMF and MLP
  - **Item2Vec** [Barkan and Koenigstein, 2016a]:
- Exploiting the equivalence between Skip-Gram and MF [Levy 2014]
  - **CoFactor**[Liang et al., 2016]: items embedding as a regularisation term
  - **LightFM** [Kula, 2015a]: neural version of BPR

Representation Learning (RL) with Embeddings

Embedding as applied to words

- Research in vector representations of words has taken off since the work of [Mikolov et al., 2013], who represent words as embedding vectors.

Representation Learning with Neural Networks

- Sparse datasets represented as low dimensional dense vectors
- Exploiting the sequential nature of user's interactions
  - **Prod2Vec** [Grobovic 2015]: next item prediction
  - **Neural Collaborative Filtering** [He et al., 2017]: A neural architecture combining GMF and MLP
  - **Item2Vec** [Barkan and Koenigstein, 2016a]:
- Exploiting the equivalence between Skip-Gram and MF [Levy 2014]
  - **CoFactor**[Liang et al., 2016]: items embedding as a regularisation term
  - **LightFM** [Kula, 2015a]: neural version of BPR

None of the previous works focus on learning user/item representation and user-preference for items jointly

Some challenges: Types of feedback

From explicit feedback...

- e.g. 5-star rating
- Pros: good quality
- Cons: scarcity

to implicit feedback

- e.g. clicking, buying
- Pros: available in abundance
- Cons: noisy, no true negatives

Some challenges: Sparsity

- A *catalog* is full database of offers (or items).
- Large number of items available in catalog and shown to the users.
- Users rate or click on only a very limited number of items, compared to what is shown.
- For data extracted from online advertising this phenomenon is even more pronounced (KASANDR and PANDOR).
- Cons: Machine learning models fail on highly sparse datasets as it is difficult to learn anything when no feedback is available from the user.

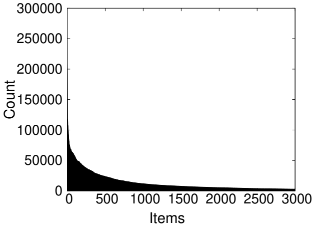
Table: An illustration of sparsity present in the datasets we used.

	# of users	# of items	# of interactions	Sparsity
ML-100K	943	1,682	100,000	**93.685%**
ML-1M	6,040	3,706	1,000,209	**95.530%**
NETFLIX	90,137	3,560	4,188,098	**98.700%**
KASANDR-GER	25,848	1,513,038	9,489,273	**99.976%**
PANDOR	1,918,968	3,755	225,579	**99.997%**

Some challenges: Popularity-Bias, Long Tail and Filter Bubble

- Popularity bias: a situation where a large majority of items have only very few ratings or clicks.

- Can lead to *"rich-get-richer-effect"* [Jannach et al., 2015]
- Limited catalog coverage (concentration bias)
- Cons: Can lead to monotonous recommendations and can harm long term retention of user's interest



- Filter Bubble [Pariser, 2011]: a phenomenon, where people do not get exposed to viewpoints different from their own

- Cons: Media favoring biased view-points for catering to user's interest.

Some challenges: Representation Learning of users and items

- Representation plays an important role in recommender systems.
- Leverage multiple sources of data for rich representation [Sonie et al., 2018].
- Representation of users and items:
  - Transactions (Example. Matrix Factorization)
  - Content: Product description, Reviews, Product image
  - User demography

How to best represent users and items

- How to take into account the transactions and contextual information in order to best represent users and items
- Cons: Bad representation can adversely affect the model's performance

Summing up contributions

- Enhance user and item representation to increase performance
- Take the Popularity bias into account
- Extract temporal topics from user and item descriptions for contextual information
- Large scale datasets containing contextual information
- Model should also:
  - scale to the data
  - work on highly sparse data
  - deal with implicit feedback

Outline

Outline

Recommending with NervE

General idea

Representation learning and pairwise ranking simultaneously using neural networks
joint work with Mikhail Trofimov, Oleg Horodnitskii, Charlotte Laclau, Yury Maximov and Massih-Reza Amini[a]

[a] Work submitted to IPM

Recommending with NervE

General idea

Representation learning and pairwise ranking simultaneously using neural networks
joint work with Mikhail Trofimov, Oleg Horodnitskii, Charlotte Laclau, Yury Maximov and Massih-Reza Amini[a]

[a]Work submitted to IPM

Notations

- $\mathcal{U}$ and $\mathcal{I}$ the set of users and items, respectively.
- **U** and **V** their latent representations
- for each $u \in \mathcal{U}$, we consider two subsets of items $\mathcal{I}_u^+, \mathcal{I}_u^- \subset \mathcal{I}$
  - i) $\mathcal{I}_u^- \neq \varnothing$ and $\mathcal{I}_u^+ \neq \varnothing$,
  - ii) for any pair $(i, i') \in \mathcal{I}_u^+ \times \mathcal{I}_u^-$; u has a preference, symbolized by $\succsim_u$. Hence $i \succsim_u i'$ implies that, user u prefers item i over item i'.

Recommending with NervE

General idea

Representation learning and pairwise ranking simultaneously using neural networks
joint work with Mikhail Trofimov, Oleg Horodnitskii, Charlotte Laclau, Yury Maximov and Massih-Reza Amini[a]

[a]Work submitted to IPM

Notations

- $\mathcal{U}$ and $\mathcal{I}$ the set of users and items, respectively.
- **U** and **V** their latent representations
- for each $u \in \mathcal{U}$, we consider two subsets of items $\mathcal{I}_u^+, \mathcal{I}_u^- \subset \mathcal{I}$

  i) $\mathcal{I}_u^- \neq \varnothing$ and $\mathcal{I}_u^+ \neq \varnothing$,

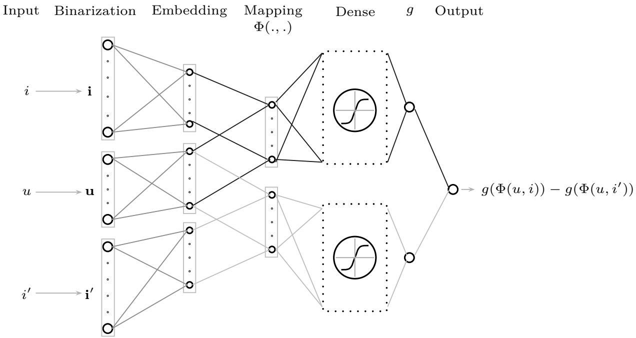
  ii) for any pair $(i, i') \in \mathcal{I}_u^+ \times \mathcal{I}_u^-$; u has a preference, symbolized by $\underset{u}{\succ}$. Hence $i \underset{u}{\succ} i'$ implies that, user u prefers item i over item i'.

Ground-truth

The desired output $y_{i,u,i'} \in \{-1, +1\}$ is defined over each triplet $(i, u, i') \in \mathcal{I}_u^+ \times \mathcal{U} \times \mathcal{I}_u^-$ as:

$$y_{i,u,i'} = \begin{cases} 1 & \text{if } i \underset{u}{\succ} i', \\ -1 & \text{otherwise.} \end{cases}$$

Graphical representation of NervE



- *Embedding* layer: transforms the sparse binary representations of the user and each of the items to denser real-valued vectors.
- *Mapping* layer: composed of two groups of units each being obtained from the element-wise product of user and item embedding
- *Dense* layer: fully connect each of these units

Objectives

- Given: a multivariate real-valued function $g(.)$
- The prediction given by NervE for an input (i, u, i') is:

$$f(i, u, i') = g(\Phi(u, i)) - g(\Phi(u, i')).$$

$$\mathcal{L}_c(f, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{z}_{i,u,i'}, y_{i,u,i'}) \in \mathcal{S}} \log(1 + e^{y_{i,u,i'}(g(\Phi(u,i')) - g(\Phi(u,i)))}).$$

$$\mathcal{L}_p(\mathbb{U}, \mathbb{V}, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{z}_{i,u,i'}, y_{i,u,i'}) \in \mathcal{S}} \Big[\log(1 + e^{y_{i,u,i'} \mathbf{U}_u^\top (\mathbf{V}_{i'} - \mathbf{V}_i)}) + \lambda(\|\mathbf{U}_u\| + \|\mathbf{V}_{i'}\| + \|\mathbf{V}_i\|) \Big]$$

NervE, then minimizes the following objective

$$\mathcal{L}_{c,p}(f, \mathbf{U}, \mathbf{V}, \mathcal{S}) = \alpha \mathcal{L}_c(f, \mathcal{S}) + (1 - \alpha) \mathcal{L}_p(\mathbf{U}, \mathbf{V}, \mathcal{S}),$$

Objectives

- Given: a multivariate real-valued function $g(.)$
- The prediction given by NervE for an input (i, u, i') is:

$$f(i, u, i') = g(\Phi(u, i)) - g(\Phi(u, i')).$$

$$\mathcal{L}_c(f, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{z}_{i,u,i'}, y_{i,u,i'}) \in \mathcal{S}} \log(1 + e^{y_{i,u,i'}(g(\Phi(u,i')) - g(\Phi(u,i))}).$$

$$\mathcal{L}_p(\mathbb{U}, \mathbb{V}, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{z}_{i,u,i'}, y_{i,u,i'}) \in \mathcal{S}} \Big[\log(1 + e^{y_{i,u,i'} \mathbf{U}_u^\top (\mathbf{V}_{i'} - \mathbf{V}_i)}) + \lambda(\|\mathbf{U}_u\| + \|\mathbf{V}_{i'}\| + \|\mathbf{V}_i\|) \Big]$$

NervE, then minimizes the following objective

$$\mathcal{L}_{c,p}(f, \mathbf{U}, \mathbf{V}, \mathcal{S}) = \alpha \mathcal{L}_c(f, \mathcal{S}) + (1 - \alpha) \mathcal{L}_p(\mathbf{U}, \mathbf{V}, \mathcal{S}),$$

Why introduce NervE?

$\mathcal{L}_{c,p}$ focuses simultaneously on both relevance of items recommended and representation of users and items
Flexible because α can be tuned to put more weight on either representation learning or preference learning

Learning objective

Scenarios over Losses

- $\mathcal{L}_c(f, \mathcal{S})$ focuses on the ability of the all architecture to respect the relative ordering of items w.r.t. user's preferences
- $\mathcal{L}_p(\mathbf{U}, \mathbf{V}, \mathcal{S})$ reflects the quality of the latent representations to respect this relative order
- $\alpha \in [0, 1]$ is a real-valued parameter to balance between ranking prediction ability and expressiveness of the learned representations
- $\mathcal{S}$ the training set consisting of triplets and ground truth.

Scenarios over α

1. $\alpha = 0$ or $\alpha = 1$
2. $\alpha = 0.5$
3. adapting the value of α at each epoch

Algorithm: Learning phase

Algorithm 1 `NervE.`: Learning phase

Input:

T: maximal number of epochs
A set of users $\mathcal{U} = \{1, \ldots, N\}$
A set of items $\mathcal{I} = \{1, \ldots, M\}$
for $ep = 1, \ldots, T$ **do**
  Randomly sample a mini-batch $\tilde{\mathcal{S}}_n \subseteq \mathcal{S}$ of size n from the original user-item matrix
  for all $((i, u, i'), y_{i,u,i'}) \in \tilde{\mathcal{S}}_n$ **do**
    Propagate (i, u, i') from the input to the output.
  Retro-propagate the pairwise ranking error estimated over $\tilde{\mathcal{S}}_n$.

Output: Users and items latent feature matrices $\mathbb{U}, \mathbb{V}$ and the model weights.

Algorithm: Testing phase

Algorithm 2 `NervE.`: Testing phase

Input:

A user $u \in \mathcal{U}$; A set of items $\mathcal{I} = \{1, \ldots, M\}$;
A set containing the k preferred items in $\mathcal{I}$ by u;
$\mathfrak{N}_{u,k} \leftarrow \varnothing$;
The output of `NervE.` learned over a training set: f
Apply f to the first two items of $\mathcal{I}$ and, note the preferred one i^* and place it at the top of $\mathfrak{N}_{u,k}$;
for $i = 3, \ldots, M$ **do**
  if $g(\Phi(u, i)) > g(\Phi(u, i^*))$ **then**
    Add i to $\mathfrak{N}_{u,k}$ at rank 1
  else
    $j \leftarrow 1$
    while $j \leq k$ AND $g(\Phi(u, i)) < g(\Phi(u, i_g)))$ // where $i_g = \mathfrak{N}_{u,k}(j)$ **do**
      $j \leftarrow j + 1$
    if $j \leq k$ **then**
      Insert i in $\mathfrak{N}_{u,k}$ at rank j

Output: $\mathfrak{N}_{u,k}$;

Introducing diversity

Motivation

- Increase the performance of a RS **on the long-term** [Zhang and Hurley, 2008, McNee et al., 2006].
- Control the **popularity bias** [Abdollahpouri et al., 2017]

Introducing diversity

Motivation

- Increase the performance of a RS **on the long-term** [Zhang and Hurley, 2008, McNee et al., 2006].
- Control the **popularity bias** [Abdollahpouri et al., 2017]

Existing approaches

- **Re-ranking** [Drosou and Pitoura, 2009, Zhang and Hurley, 2008].
- **Clustering** of items [Li and Murata, 2012, Shi, 2013].
- **Multi-function optimization** [Su et al., 2013, Wasilewski and Hurley, 2016].
- Using **structured learning** [Cheng et al., 2017]

Introducing diversity

Motivation

- Increase the performance of a RS **on the long-term** [Zhang and Hurley, 2008, McNee et al., 2006].
- Control the **popularity bias** [Abdollahpouri et al., 2017]

Existing approaches

- **Re-ranking** [Drosou and Pitoura, 2009, Zhang and Hurley, 2008].
- **Clustering** of items [Li and Murata, 2012, Shi, 2013].
- **Multi-function optimization** [Su et al., 2013, Wasilewski and Hurley, 2016].
- Using **structured learning** [Cheng et al., 2017]

Challenge: finding a trade-off between relevance and diversity and to introduce diversity when there is no meta-information available about items

KL Divergence in $\texttt{NervE}_c$

- $\texttt{NervE}_c$ is the version of $\texttt{NervE}$ with L_c.
- In order to incorporate diversity in $\texttt{NervE}_c$, we introduce KL-Divergence term in $\texttt{NervE}_c$:

$$\mathcal{L}_{\texttt{NervE}_c}(f, U, V, \mathcal{S}) + \boxed{\beta \frac{1}{|U|} \sum_{u \in U} \left(\frac{1}{k(k-1)} \right) \sum_{i,i' \in \mathcal{S}_u^k} KL(\mathbf{V}_i^{\ell_1} || \mathbf{V}_{i'}^{\ell_1})},$$

- $\mathbf{V}_i^{\ell_1}$ (resp. $\mathbf{V}_{i'}^{\ell_1}$) is the ℓ_1-normalized embedding associated with item i (resp. i')
- Embeddings computed using Item2Vec [Barkan and Koenigstein, 2016b]
- β is the diversity inducing regularization parameter
- By inducing diversity in relevance function, we are able to overcome popularity bias (as we will see in experiments section)

KL Divergence in $\texttt{NervE}_c$

- $\texttt{NervE}_c$ is the version of $\texttt{NervE}$ with L_c.
- In order to incorporate diversity in $\texttt{NervE}_c$, we introduce KL-Divergence term in $\texttt{NervE}_c$:

$$\mathcal{L}_{\texttt{NervE}_c}(f, U, V, \mathcal{S}) + \boxed{\beta \frac{1}{|U|} \sum_{u \in U} \left(\frac{1}{k(k-1)} \right) \sum_{i, i' \in \mathcal{S}_u^k} KL(\mathbf{V}_i^{\ell_1} || \mathbf{V}_{i'}^{\ell_1})},$$

- $\mathbf{V}_i^{\ell_1}$ (resp. $\mathbf{V}_{i'}^{\ell_1}$) is the ℓ_1-normalized embedding associated with item i (resp. i')
- Embeddings computed using Item2Vec [Barkan and Koenigstein, 2016b]
- β is the diversity inducing regularization parameter
- By inducing diversity in relevance function, we are able to overcome popularity bias (as we will see in experiments section)

Diversity with no meta-information

Introducing diversity, via embeddings learned from interactions when no meta-information about items is available not addressed by previous works

Summing up

1st Contribution:

- Proposed a NN based collaborative ranking model that learns a hybrid loss function which focuses on both representation learning and pairwise learning-to-rank function.
- NN that paves the way for introducing diversity in a single step of learning process
- Diversity induced in NervE using item embedding
- NN that jointly learns the representation and the ranking of items

What was not presented:

- To take into account contextual features
- Can deal with both item and user cold-start by taking into account contextual features
- Theoretical analysis

Outline

Motivation: Why make use of temporal topics

Prevalence of textual data in RS

- Lot of textual data around; example Amazon reviews, News recommendation systems.
- Topic modeling is a good way to exploit textual data

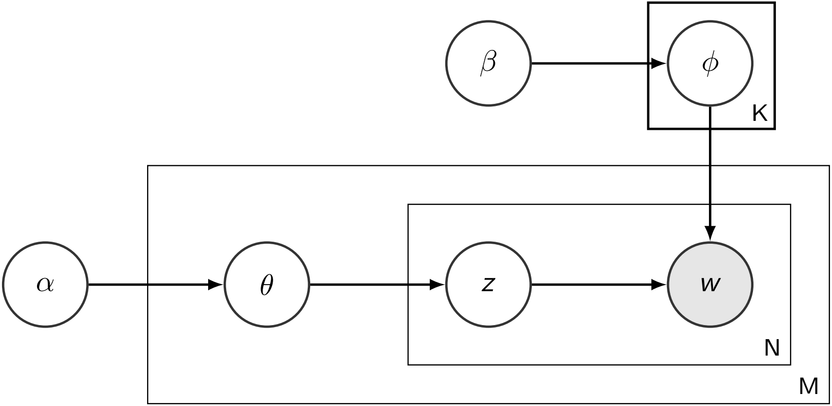
Importance of time

- Recommendations are actions at a moment in time ; Time is a critical aspect in any RS [Basilico and Raimond, 2017]

How to use it?

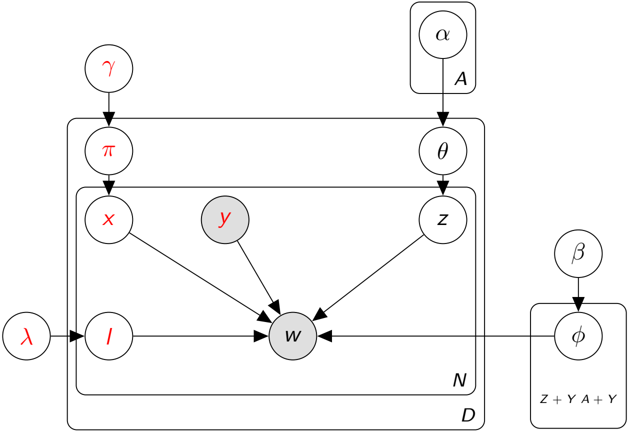
- RS data are usually sparse with users providing very little feedback about their preferences
- **Temporal** topics inferred from text can be used as **supplementary feature** in RS models.
- This contextual information of topics can also be used in content-filtering based RS

General-purpose topic modelling with LDA



- Given a set of documents, the goal of LDA is to come up with underlying topic distributions of the documents

General-purpose topic modelling with TAM



- An *aspect* is defined as a characteristic that spans the document such as an underlying theme.

Time-Aware Topic Aspect Model

→Random variables a and t in TAM[1]

Document level characteristic a

- TAM can then be extended to include a document level characteristic a.
- Document level characteristic could be anything such as *overall sentiment*.
- Sentiment can be positive, negative or could contain more values.

Evolution of sentiment with time

- Sentiment towards topics also evolves with time.
- Taste of users towards various topics keep changing with time.

Introduction of random variable t in TAM

- We introduce a random variable t for time in TAM.
- a (sentiment) is drawn depending on time t.
- Time t itself is drawn from a multinomial distribution ψ.

[1] Sidana et al., IEEE TKDE'18

Plate diagram of time-aware TAM

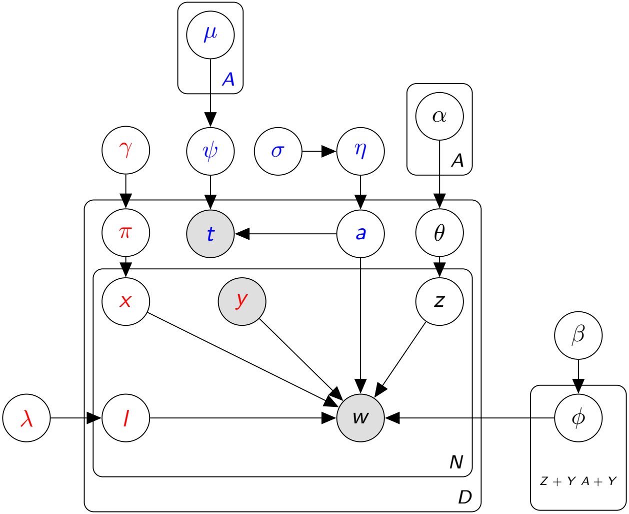


Figure: Time-Aware Topic Aspect Model. Sentiment *a* is time-aware.

TM–LDA

- Modeling evolution of topics of dynamic collection of documents over time.
- At the heart of the algorithm lies the following equation.

$$\theta_i \approx \frac{\theta_{i-1}.M}{\|\theta_{i-1}.M\|_{\ell_1}} \quad (1)$$

- M is a $k \times k$ matrix, called the transition matrix, and k is the number of topics.
- Solution:

$$M = \arg\min_X \|A.X - B\|_F \quad (2)$$

- TM–LDA is quite elegant in modeling general purpose topics over time
- Cons: Huge amount of postprocessing to model transition matrices

Summing up

What was presented:

- General purpose topic models like LDA and TAM
- Introduction of two random variables in TAM: t and a
- An existing temporal topic model TM–LDA, which we will use later to enrich feature set in order to improvise recommendations

What was not presented:

- Application of TAM with random variables t and a on health-monitoring social-media dataset

Outline

KASANDR: Collection of the data

KASANDR[a] (Kelkoo lArge ScAle juNe Data for Recommendation)

[a]Sidana et al., *KASANDR: A Large-Scale Dataset with Implicit Feedback for Recommendation*, SIGIR'17

- Records interactions of Kelkoo's customers from 20 countries in June
- 4 services: Ads, Kelkoo's Website, Partners, Kelkoo Feed System

Different scenarios in which the database gets populated

- User visits Kelkoo's website and enters a search keyword
- User browsing through Kelkoo's or partner's website is shown an ad
- User enters search keywords in Kelkoo's partner's website

What is new about KASANDR

- Rich set of contextual feautres (users and items)
- First implicit feedback recommender dataset of this size contributed and made public by any research lab (Uncompressed size: 950 GB.)

KASANDR: Description

General Statistics

- KASANDR comes with meta-information about offers and users
  - Users : country code
  - Items : product category, price, query string

# of users	# of unique offers	# of offers shown	# of clicks	Sparsity
123,529,420	56,667,919	3,210,050,267	16,107,227	99.9999997848%

- Users are shown **26 offers** in average
- Average number of clicks is only **1.71**, for users with at least one click

KASANDR: Description

General Statistics

- KASANDR comes with meta-information about offers and users
  - Users : country code
  - Items : product category, price, query string

# of users	# of unique offers	# of offers shown	# of clicks	Sparsity
123,529,420	56,667,919	3,210,050,267	16,107,227	99.9999997848%

- Users are shown **26 offers** in average
- Average number of clicks is only **1.71**, for users with at least one click

Time-granularity and Cold Start

- Few number of users return to the system.
- Significance of taking *time-granularity*
- Significance of handling user cold-start

Week Number	# New Users	# Returning Users
23	36,932,009	165,951
24	26,736,201	199,467
25	22,358,876	185,749
26	13,908,242	135,303

PANDOR: Collection of the data

PANDOR (Purch dAta for oNline recommeDation and cOld-staRt) [a]

[a]Sidana et al., *Learning to recommend diverse items over implicit feedback on PANDOR*, RecSys'18

- User's traffic collected from Tom's hardware website
- User browsing through Purch's websites is shown an ad
- Another implicit feedback recommender system data set belonging to Purch's one month click logs made public

What is new about PANDOR?

- First dataset to contain actual English text words and information about page text and offer text
- Page text on which offer was displayed
- No previous datasets contain actual text words

PANDOR basic statistics

- Overall Dataset Statistics, for 1 month.

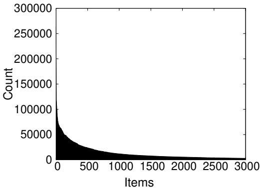
# of users	# of unique offers	# of offers shown	# of clicks	Sparsity
5,894,431	14,716	48,754,927	337,511	99.99961

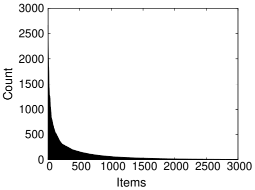
- PANDOR provides a rich set of textual information
  - offer text
  - page text
  - keywords

Page text vocabulary size	9,111
Product text vocabulary size	6,016
Keyword vocabulary size	543
# Offers which have at least 1 text word	2,701 (27.4%)
# Pages which have at least 1 text word	1,990 (28.1%)

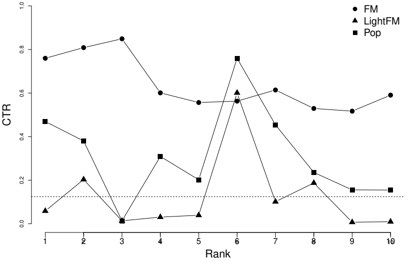
PANDOR popularity bias

- **Long tail item** : number of time an item is recommended and clicked.





- **Rank** of items as a function of **CTR**
- Strong ML approaches recommend
  - popular items
  - items with high CTR
- Why? Current RS system at Purch is based on **popularity**



Outline

Evaluation: Expected Intra List Diversity

→ Relevance Evaluation: Mean Average Precision@different ranks

- Intra-List distance of any list $L(s)$ of items recommended to a particular user is given by:

$$ILD(L_u) = \frac{1}{N(N-1)} \sum_{i,j \in L(s)} d(i,j) \tag{3}$$

- EILD is then given by averaging over all users:

$$EILD = \frac{1}{|U|} \sum_{u \in U} ILD \tag{4}$$

- Distance $d(i,j)$ between two items i and j is computed using meta-data of items suchas item-genre, item-category or item-embeddings.
- High value of EILD indicates high diversity.

Evaluation Baselines

Non-Machine Learning

- The random rule (Rand)
- The popularity rule (Pop)
- The past interaction technique (PastI)

Classification-Based

- Matrix Factorization (MF) [Koren et al., 2009].
- Factorization Machines (FM) [Rendle, 2010].
- Field-Aware Factorization Machines (FFM) [Juan et al., 2016]

Ranking-Based

- Rank-ALS [Takács and Tikk, 2012]
- Bayesian Personalized Ranking (BPR) [Rendle et al., 2009]
- LightFM [Kula, 2015b],
- `Co-Factor` [Liang et al., 2016] co-occurrence counts.
- $\texttt{NervE}_{c,p}$ uses a linear combination of $\mathcal{L}_p$ and $\mathcal{L}_c$.
- $\texttt{NervE}_p$ focuses on the quality of the latent representation of users and items
- $\texttt{NervE}_c$ focuses on the accuracy of the score at the output of the framework

Datasets

- We use following datasets to evaluate the performance of `NervE`
  - ML-100K
  - ML-1M
  - NETFLIX
  - KASANDR-GER

	# of users	# of items	# of interactions	Sparsity
ML-100K	943	1,682	100,000	93.685%
ML-1M	6,040	3,706	1,000,209	95.530%
NETFLIX	90,137	3,560	4,188,098	98.700%
KASANDR-GER	25,848	1,513,038	9,489,273	99.976%

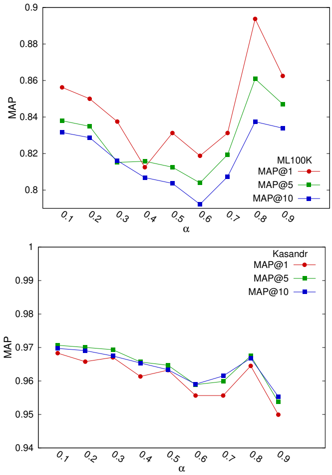
- Statistics of various collections used in our experiments after preprocessing.
- KASANDR-GER is the only true implicit feedback, others are synthetically made implicit

Evaluation Settings

- We consider two settings w.r.t. to the set of items selected for the prediction.
  - Item recommendation only relies on past interacted offers (shown + clicked) ; arguably the most common in academic research
  - Item recommendation only relies on all offers ; reflects this real-world scenario as, at the time of making the recommendation, the notion of shown items is not available
- We use Mean Average Precision (MAP) for evaluating relevance of recommended list of offers
- For all datasets, we only keep users who have rated at least five movies and remove users who gave the same rating for all movies.
- In addition, for NETFLIX, we take a subset of the original data and randomly sample 20% of the users and 20% of the items.

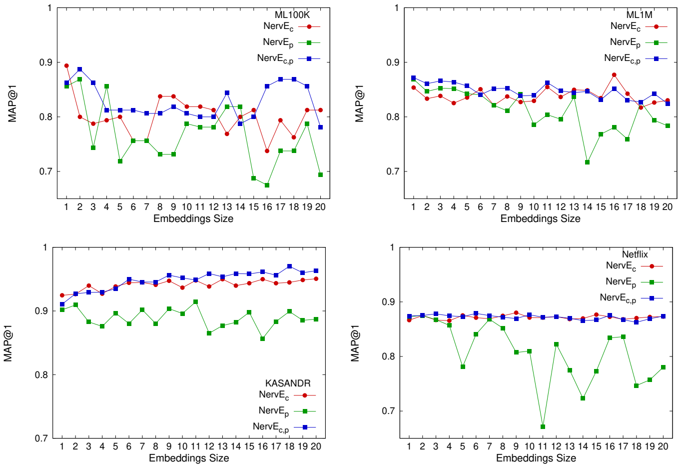
NervE performance with changing the α parameter

- α is the tradeoff between representation learning and preference score learning



NervE performance with changing the embeddings

- Best MAP@1 results are generally obtained with small sizes of item and user embedded vector spaces k



Parameter Tuning

	ML-100K			ML-1M			Netflix			Kasandr-Ger		
	NervE_c	NervE_p	$\mathrm{NervE}_{c,p}$	NervE_c	NervE_p	$\mathrm{NervE}_{c,p}$	NervE_c	NervE_p	$\mathrm{NervE}_{c,p}$	NervE_c	NervE_p	$\mathrm{NervE}_{c,p}$
k	1	2	2	16	1	1	9	2	6	19	1	18
λ	0.05	0.005	0.005	0.05	0.0001	0.001	0.05	0.01	0.05	0.0001	0.05	0.005
# units	32	64	16	32	16	32	64	16	16	64	16	64

- Best parameters for NervE_p, NervE_c and $\mathrm{NervE}_{c,p}$ when prediction is done on only shown offers.
- k denotes the dimension of embeddings.
- λ the regularization parameter.
- We also report the number of hidden units per layer.

	ML-100K			ML-1M			Netflix			Kasandr-Ger		
	NervE_c	NervE_p	$\mathrm{NervE}_{c,p}$	NervE_c	NervE_p	$\mathrm{NervE}_{c,p}$	NervE_c	NervE_p	$\mathrm{NervE}_{c,p}$	NervE_c	NervE_p	$\mathrm{NervE}_{c,p}$
k	15	5	8	2	11	2	3	13	1	4	16	14
λ	0.001	0.001	0.001	0.05	0.0001	0.001	0.0001	0.001	0.001	0.001	0.0001	0.05
# units	32	16	16	32	64	32	32	64	64	32	64	64

- Best parameters for NervE_p, NervE_c and $\mathrm{NervE}_{c,p}$ when prediction is done on all offers.

Results on Interacted Offers

	ML-100K		ML-1M		NETFLIX		KASANDR-GER	
	MAP@1	MAP@10	MAP@1	MAP@10	MAP@1	MAP@10	MAP@1	MAP@10
BPR-MF	0.613$^{\downarrow}$	0.608$^{\downarrow}$	0.788$^{\downarrow}$	0.748$^{\downarrow}$	**0.909**	0.842$^{\downarrow}$	0.857$^{\downarrow}$	0.857$^{\downarrow}$
LightFM	0.772$^{\downarrow}$	0.770$^{\downarrow}$	0.832$^{\downarrow}$	0.795$^{\downarrow}$	0.800$^{\downarrow}$	0.793$^{\downarrow}$	0.937$^{\downarrow}$	0.936$^{\downarrow}$
CoFactor	0.718$^{\downarrow}$	0.716$^{\downarrow}$	0.783$^{\downarrow}$	0.741$^{\downarrow}$	0.693$^{\downarrow}$	0.705$^{\downarrow}$	0.925$^{\downarrow}$	0.918$^{\downarrow}$
NervE$_c$	**0.894**	**0.848**	0.877$^{\downarrow}$	0.835	0.880$^{\downarrow}$	**0.847**	0.958$^{\downarrow}$	0.963$^{\downarrow}$
NervE$_p$	0.881$^{\downarrow}$	0.846	0.876$^{\downarrow}$	**0.839**	0.875$^{\downarrow}$	0.844	0.915$^{\downarrow}$	0.923$^{\downarrow}$
NervE$_{c,p}$	0.888$^{\downarrow}$	0.842	**0.884**	**0.839**	0.879$^{\downarrow}$	**0.847**	**0.970**	**0.973**

- Prediction is done only on offers shown to users. The best result is in bold.
- Beats all the other algorithms on KASANDR-GER, ML-100K and ML-1M.
- On NETFLIX, BPR-MF outperforms our approach in terms of MAP@1.

Results on All Offers

	ML-100K		ML-1M		NETFLIX		KASANDR-GER	
	MAP@1	MAP@10	MAP@1	MAP@10	MAP@1	MAP@10	MAP@1	MAP@10
BPR-MF	0.140$^{\downarrow}$	**0.261**	0.048$^{\downarrow}$	0.097$^{\downarrow}$	0.035$^{\downarrow}$	0.072$^{\downarrow}$	0.016$^{\downarrow}$	0.024$^{\downarrow}$
LightFM	0.144$^{\downarrow}$	0.173$^{\downarrow}$	0.028$^{\downarrow}$	0.096$^{\downarrow}$	0.006$^{\downarrow}$	0.032$^{\downarrow}$	0.002$^{\downarrow}$	0.003$^{\downarrow}$
CoFactor	0.056$^{\downarrow}$	0.031$^{\downarrow}$	0.089$^{\downarrow}$	0.033$^{\downarrow}$	0.049$^{\downarrow}$	0.030$^{\downarrow}$	0.002$^{\downarrow}$	0.001$^{\downarrow}$
$\texttt{NervE}_c$	0.106$^{\downarrow}$	0.137$^{\downarrow}$	0.067$^{\downarrow}$	0.093$^{\downarrow}$	0.032$^{\downarrow}$	0.048$^{\downarrow}$	0.049$^{\downarrow}$	0.059$^{\downarrow}$
$\texttt{NervE}_p$	**0.239**	0.249	**0.209**	**0.220**	**0.080**	**0.089**	0.100$^{\downarrow}$	0.100$^{\downarrow}$
$\texttt{NervE}_{c,p}$	0.111$^{\downarrow}$	0.134$^{\downarrow}$	0.098$^{\downarrow}$	0.119$^{\downarrow}$	0.066$^{\downarrow}$	0.087	**0.269**	**0.284**

- Prediction is done on all offers. The best result is in bold
- All the algorithms encounter an extreme drop of their performance in terms of MAP
- NervE framework significantly outperforms all other algorithms
- This difference is all the more important on KASANDR-GER

Comparison between NervE versions

	ML-100K		ML-1M		Netflix		Kasandr-Ger	
	MAP@1	MAP@10	MAP@1	MAP@10	MAP@1	MAP@10	MAP@1	MAP@10
$\texttt{NervE}_c$	0.106$^\downarrow$	0.137$^\downarrow$	0.067$^\downarrow$	0.093$^\downarrow$	0.032$^\downarrow$	0.048$^\downarrow$	0.049$^\downarrow$	0.059$^\downarrow$
$\texttt{NervE}_p$	**0.239**	0.249	**0.209**	**0.220**	**0.080**	**0.089**	0.100$^\downarrow$	0.100$^\downarrow$
$\texttt{NervE}_{c,p}$	0.111$^\downarrow$	0.134$^\downarrow$	0.098$^\downarrow$	0.119$^\downarrow$	0.066$^\downarrow$	0.087	**0.269**	**0.284**

	ML-100K		ML-1M		Netflix		Kasandr-Ger	
	MAP@1	MAP@10	MAP@1	MAP@10	MAP@1	MAP@10	MAP@1	MAP@10
$\texttt{NervE}_c$	**0.894**	**0.848**	0.877$^\downarrow$	0.835	0.880$^\downarrow$	**0.847**	0.958$^\downarrow$	0.963$^\downarrow$
$\texttt{NervE}_p$	0.881$^\downarrow$	0.846	0.876$^\downarrow$	**0.839**	0.875$^\downarrow$	0.844	0.915$^\downarrow$	0.923$^\downarrow$
$\texttt{NervE}_{c,p}$	0.888$^\downarrow$	0.842	**0.884**	**0.839**	0.879$^\downarrow$	**0.847**	**0.970**	**0.973**

- ($\texttt{NervE}_{c,p}$) increases the quality of overall recommendations
- ($\texttt{NervE}_{c,p}$) outperforms ($\texttt{NervE}_p$) and ($\texttt{NervE}_c$) on ML-1M, Kasandr-Ger and Netflix (interacted offers setting).
- ($\texttt{NervE}_{c,p}$) outperforms ($\texttt{NervE}_p$) and ($\texttt{NervE}_c$) on Kasandr-Ger.
- Optimizing both losses simultaneously is beneficial in case of true implicit feedback datasets such as Kasandr-Ger(all offers setting)
- In case of interacted offers setting, optimizing ranking and embedding loss simultaneously boosts performance on all datasets.

Outline

Evaluation Setting on PANDOR

Data preprocessing

- 1,767,589 interactions from 119,536 unique users on 2,840 unique items.
- Sort all interactions according to time; filter out users without a single click; take the first 70% interactions for training and the remaining 30% for testing.

Prediction settings

- We consider both settings w.r.t. to the set of items selected for the prediction.
- For the first setting, the prediction is done over 20.653 items on average
- For the second one, the prediction is over 2840 items.

Baselines

- Baselines: Rank-ALS, BPR-MF, FM, LightFM
- As PANDOR suffers from popularity bias, we show diversity results.

Results on PANDOR

	MAP@1	MAP@5	MAP@10	EILD@10
Random	0.135	0.157	0.161	0.172
Popularity	0.249	0.262	0.266	0.080
FM (SGD)	0.244	0.269	0.273	**0.191**
BPR-MF	0.222	0.240	0.229	0.173
LightFM	**0.479**	**0.526**	**0.535**	0.099
$\texttt{NervE}_C$	0.251	0.292	0.299	0.115
Rank-ALS	0.256	0.261	0.261	0.008

- MAP@k obtained for all compared approaches on interacted items on PANDOR. The best results are in bold.

	MAP@1	MAP@5	MAP@10	EILD@10
Random	9.934e-05	0.0001	0.0001	0.536
Popularity	**0.007**	**0.009**	**0.011**	0.396
FM (SGD)	0.001	0.002	0.003	0.534
BPR-MF	0.005	0.008	0.010	0.493
LightFM	0.0002	0.0008	0.002	0.287
$\texttt{NervE}_C$	0.006	0.008	0.010	**0.560**
Rank-ALS	0.002	0.002	0.003	0.564

- MAP@k obtained for all compared approaches on all items on PANDOR.

Introduction of Diversity on PANDOR

- We explore the ability of diversity in RS to overcome the strong bias induced by popular items.
- we focus only on the setting in which we test on all items as most approaches fail to provide good results on such setting.
- Two approaches tested:
  - $\texttt{NervE}_c$
  - RankALS [Wasilewski and Hurley, 2016]
- The first one was initially proposed by [Wasilewski and Hurley, 2016]
- We compute item embeddings, with Gensim based Skip-Gram implementation of Word2Vec
- Dimension of embeddings: 20 and the context window: 3.

Diversity Results on PANDOR

Metric Maximized	β	MAP@10	EILD@10
MAP@10	0.0001	0.010	0.633
EILD	0.1	0.001	0.666
HM(MAP@10,EILD)	0.0001	0.010	0.633
HM(MAP@10,EILD) while maximizing diversity	-0.75	0.006	0.635

- Results of $\texttt{NervE}_c$ coupled with diversity.

Metric Maximized	*regularizer*	MAP@10	EILD@10
MAP@10	PLapDQ-min	0.018	0.552
EILD	No-Regularizer	0.0002	0.692
HM(MAP@10,EILD)	PLapDQ-min	0.018	0.552
HM(MAP@10,EILD) while maximizing diversity	DQ-max	0.016	0.553

- Results of RankALS coupled with diversity.

	Before Diversity				After Diversity			
	MAP@1	MAP@5	MAP@10	EILD@10	MAP@1	MAP@5	MAP@10	EILD@10
$\texttt{NervE}_C$	0.006	0.008	0.010	0.561	0.009	0.009	0.010	0.633
RankALS	0.002	0.002	0.003	0.564	0.010	0.014	0.016	0.553

- By Introducing diversity we are able to increase both relevance of the items and diversity of items

Topic-Modelling application to RS on PANDOR

Model and features used

- We use one of the topic models, namely TM–LDA, and feed the topics derived from it, as contextual information to Factorization Machines.
- We use PANDOR page textual information because pages, which users are browsing on, depict the interest of the user

Data Preprocessing

- We first sort the dataset temporally and remove all the users who did not do a single click
- We, then, take first 80% for training and remaining 20% for testing.

- MAP@k improves after putting TM–LDA-based topics as contextual information in Factorization Machines on interacted items on PANDOR.

	MAP@1	MAP@5	MAP@10
Random	0.135	0.157	0.161
Popularity	0.249	0.262	0.266
Factorization Machines (FM)	0.244	0.269	0.273
TM–LDA-Based FM using Page Text	**0.385**(57.7%↑)	**0.390**(45%↑)	**0.389**(42.5%↑)

Outline

Main contributions

- A neural network, named `NervE` to learn user preference and representation for implicit feedback simultaneously
- An enormous data set, named KASANDR, consisting of clicks containing rich meta information
- A data set, named PANDOR, consisting of rich text and affected by popularity bias facilitating research on algorithms, which help overcome the same.
- Introduction of diversity in loss function of $\texttt{NervE}_c$ using KL-Divergence and item embeddings
- Introduction of 2 novel topic models to make use of contextual information in recommender algorithms

Future directions

- Use contextual information (meta-information), topics in `NervE`.
- Use time-aware topic models to do content-based recommendations.
- Do online evaluation and A/B testing of `NervE` and time-aware topic models.
- Learning automatically α in `NervE`.
- Taking into account the imbalance between types of feedback.

Thank you for your attention!

References I

Abdollahpouri, H., Burke, R., and Mobasher, B. (2017).
Controlling popularity bias in learning-to-rank recommendation.
In *Proceedings of RecSys*, pages 42–46, New York, NY, USA. ACM.

Barkan, O. and Koenigstein, N. (2016a).
Item2vec: Neural item embedding for collaborative filtering.
In *Proceedings of the Poster Track of RecSys*.

Barkan, O. and Koenigstein, N. (2016b).
ITEM2VEC: neural item embedding for collaborative filtering.
In *International Workshop on Machine Learning for Signal Processing, MLSP*, pages 1–6.

Basilico, J. and Raimond, Y. (2017).
Déjà vu: The importance of time and causality in recommender systems.
In *Proceedings of RecSys*, page 342. ACM.

References II

Cheng, P., Wang, S., Ma, J., Sun, J., and Xiong, H. (2017).
Learning to recommend accurate and diverse items.
In *Proceedings of the 26th International Conference on World Wide Web, WWW '17*.

Drosou, M. and Pitoura, E. (2009).
Diversity over continuous data.
IEEE Data Eng. Bull., 32(4):49–56.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T. (2017).
Neural collaborative filtering.
In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 173–182.

Jannach, D., Lerche, L., Kamehkhosh, I., and Jugovac, M. (2015).
What recommenders recommend: an analysis of recommendation biases and possible countermeasures.
User Model. User-Adapt. Interact., 25(5):427–491.

References III

Juan, Y., Zhuang, Y., Chin, W., and Lin, C. (2016).
Field-aware factorization machines for CTR prediction.
In *Proceedings of RecSys*, pages 43–50.

Koren, Y., Bell, R. M., and Volinsky, C. (2009).
Matrix factorization techniques for recommender systems.
IEEE Computer, 42(8):30–37.

Kula, M. (2015a).
Metadata embeddings for user and item cold-start recommendations.
In Bogers, T. and Koolen, M., editors, *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015.*, volume 1448 of *CEUR Workshop Proceedings*, pages 14–21. CEUR-WS.org.

References IV

Kula, M. (2015b).
Metadata embeddings for user and item cold-start recommendations.
In *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with RecSys.*, pages 14–21.

Li, X. and Murata, T. (2012).
Using multidimensional clustering based collaborative filtering approach improving recommendation diversity.
In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Macau, China, December 4-7, 2012*, pages 169–174.

Liang, D., Altosaar, J., Charlin, L., and Blei, D. M. (2016).
Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence.
In *Proceedings of RecSys*, pages 59–66.

References V

McNee, S. M., Riedl, J., and Konstan, J. A. (2006).
Being accurate is not enough: how accuracy metrics have hurt recommender systems.
In *Extended Abstracts Proceedings of the Conference on Human Factors in Computing Systems, CHI*, pages 1097–1101.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013).
Efficient estimation of word representations in vector space.
CoRR, abs/1301.3781.

Pariser, E. (2011).
The Filter Bubble: What the Internet Is Hiding from You.
Penguin Group , The.

Rendle, S. (2010).
Factorization machines.
In *Proceedings of ICDM*, pages 995–1000.

References VI

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009).
BPR: bayesian personalized ranking from implicit feedback.
In *Proceedings of UAI*, pages 452–461.

Shi, L. (2013).
Trading-off among accuracy, similarity, diversity, and long-tail: a graph-based recommendation approach.
In *Proceedings of RecSys*, pages 57–64.

Sonie, O., Sarkar, S., and Kumar, S. (2018).
Concept to code: learning distributed representation of heterogeneous sources for recommendation.
In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 531–532.

Su, R., Yin, L., Chen, K., and Yu, Y. (2013).
Set-oriented personalized ranking for diversified top-n recommendation.
In *Proceedings of RecSys*, pages 415–418.

References VII

Takács, G. and Tikk, D. (2012).
Alternating least squares for personalized ranking.
In *Proceedings of RecSys*, pages 83–90.

Wasilewski, J. and Hurley, N. (2016).
Incorporating diversity in a learning to rank recommender system.
In *Proceedings of FLAIRS*, pages 572–578.

Zhang, M. and Hurley, N. (2008).
Avoiding monotony: improving the diversity of recommendation lists.
In *Proceedings of RecSys*, pages 123–130.