Dynamic Bayesian Metric Learning for Personalized Product Search

Teng Xiao*
School of Data and Computer Science, Sun Yat-sen University, China
tengxiao01@gmail.com

Jiaxin Ren*
School of Data and Computer Science, Sun Yat-sen University, China
renjx@mail2.sysu.edu.cn

Zaiqiao Meng
School of Computing Science, University of Glasgow, Scotland, UK
zaiqiao.meng@gmail.com

Huan Sun
Department of Computer Science and Engineering, The Ohio State University, U.S
sun.397@osu.edu

Shangsong Liang†
School of Data and Computer Science, Sun Yat-sen University, China
liangshangsong@gmail.com

ABSTRACT

In this paper, we study the problem of personalized product search under streaming scenarios. We address the problem by proposing a **D**ynamic **B**ayesian **M**etric **L**earning model, abbreviated as DBML, which can collaboratively track the evolutions of latent semantic representations of different categories of entities (i.e., users, products and words) over time in a joint metric space. In particular, unlike previous work using inner-product metric to model the affinities between entities, our DBML is a novel probabilistic metric learning approach that is able to avoid the contradicts, keep the triangle inequality in the latent space, and correctly utilize implicit feedbacks. For inferring dynamic embeddings of the entities, we propose a scalable online inference algorithm, which can jointly learn the latent representations of entities and smooth their changes across time, based on amortized inference. The inferred dynamic semantic representations of entities collaboratively inferred in a unified form by our DBML can benefit not only for improving personalized product search, but also for capturing the affinities between users, products and words. Experimental results on large datasets over a number of applications demonstrate that our DBML outperforms the state-of-the-art algorithms, and can effectively capture the evolutions of semantic representations of different categories of entities over time.

CCS CONCEPTS

• **Information systems** → *Personalization*;

KEYWORDS

Product search; Metric learning; Online learning; Probabilistic model

*Teng Xiao and Jiaxin Ren contributed equally as co-first authors.
†Shangsong Liang is the corresponding author.

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3358057

ACM Reference Format:
Teng Xiao, Jiaxin Ren, Zaiqiao Meng, Huan Sun, and Shangsong Liang. 2019. Dynamic Bayesian Metric Learning for Personalized Product Search. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357384.3358057

1 INTRODUCTION

Product search [1, 11, 15, 18, 19, 22, 41] has been receiving significant attention in academia and industry as the popularity of E-shopping increases. This is, perhaps, one of the most active behaviors on the internet right now. Product search aims at retrieving a ranked list of products according to their relevance to the query. In this paper, we study the problem of personalized product search [1, 18, 19], the goal of which is to return a ranked list of products that should be relevant to the input query and cater for users' diverse interests underlying their shopping behaviors.

Recently, various personalized product search algorithms [1, 15, 18, 19] have been proposed. However, they still suffer from a number of defects: (1) Previous work [1, 19] mainly considers static scenario. However, users' preferences, popularities of products and semantics of words in reviews evolve over time. For instance, the product may go out of fashion because of better alternatives, users' personal interests may shift from one fashion to another, and words in generated reviews by users may have different semantics over times. Static personalized product search algorithms can not explicitly capture the semantic similarities among users, products and words, which thus leads to poor product search performance. (2) Previous work [1, 41] are mainly based on Latent Vector Space Models [41], which apply matrix factorization to model relationships between users, products and words. As revealed in [21, 40], matrix factorization applied in the models potentially causes a number of problems such as contradictions, violating triangle inequality (see 4.1 for details). (3) Very recent work [15, 18, 44] try to utilize Recurrent Neural Networks (RNNs) to model evolutions of users' preferences. However, they can not explicitly and jointly model the evolutions of products and words. Besides, the RNNs they work with also bring some issues: they only make predictions in the original observation space, and don't learn a compressed representation of data [17]. As a result, these models tend to be computationally heavy and are lack of interpretability. Yet, the interpretability is

the key to explore users' dynamic behaviors and products' popularity. (4) All of them represent the entities, i.e., users, products and words, simply as points in a low dimensional space, failing to capture the uncertainties of them, which is, however, crucial in product search and other applications, e.g., decision making [14] and representation learning [5, 31].

To address the aforementioned defects, we propose **D**ynamic **B**ayesian **M**etric **L**earning, abbreviated as DBML. Our model considers personalized product search in a streaming scenario, where users' search and review comments arrive sequentially over time. DBML can collaboratively and explicitly capture the evolution of users, products and words over time in a joint metric space, such that the affinities between them can be effectively measured over time and avoid the problems in traditional matrix factorization. Different from RNN-based models [15, 18], the proposed DBML can be seen as a non-linear state-space model (SSM) [42] and the learned dynamic latent representations in the metric space are highly interpretable, and hence our DBML is able to dynamically discover products' popularities, distinguish different types of products, and discover users' behavior over time. Moreover, due to the inherent Bayesian nature of DBML, it can learn posterior Gaussian distributions of latent representations of the entities. The Gaussian distributions can capture uncertainty of latent representations and further enhance the product search performance.

The main contributions of this paper can be summarized as: (a) We propose DBML, a novel personalized product search model under streaming scenario, which can explicitly and collaboratively learn representations of different categories of entities in a joint metric space over time. To the best of our knowledge, our DBML is the first dynamic learning model that learns latent representations via a unified metric learning framework. (b) As DBML learns representations in a joint metric space, the learned representations have highly interpretability and can capture uncertainty of entities, which are important to discover latent patterns. (c) Based on amortized inference, we develop two efficient inference algorithms for variational Bayesian inference, i.e., online and offline inference. Leveraging online and offline inference algorithms, DBML can strike a balance between accuracy and efficiency for dynamic product search. (d) Experimental results on four Amazon Review datasets show DBML can achieve much better product search performance over the state-of-the-art algorithms, track the evolutions of latent representations of users, products and words over time.

2 RELATED WORK

There are three lines of research related to our work: product search, metric learning and dynamic embedding models.

2.1 Product Search

Duan et al. [12] propose a probabilistic product entity retrieval model to rank products based on query generation. To further consider the critical information from the entity space and the connections between queries and products, Duan and Zhai [11] also propose to learn user intent from query and product space. As revealed in [41], there are semantic mismatch problem existing in product search. Van Gysel et al. [41] propose a latent vector space model to jointly learn representations of words and products.All the methods above only consider the relevance between products and queries. Yet, it is critical to consider users' personalized preferences for products and search behaviors. Ai et al. [1] introduce a hierarchical embedding model to personalized product search. Recent studies [15, 18] consider the dynamics of users' preferences and utilize RNNs (Recurrent Neural Networks) to capture the dynamics of users' preferences. In this work, instead of using RNNs, we propose a dynamic embedding algorithm to explicitly and jointly model the dynamic processes of users' preferences, popularities of products and semantics of words in reviews evolving over time in a joint metric space.

2.2 Metric Learning

The idea of metric learning was first introduced for nearest neighbor classification [38], where the pair-wise similarity of vectors was modeled in a metric space. Due to the powerful representative abilities, it has achieved achievements in various fields, e.g., image classification [34] and text retrieval [25]. Comprehensive survey can be found in [3]. Recently, Collaborative Metric Learning (CML) [21] shows metric learning can satisfy the crucial triangle inequality which makes it able to capture fine-grained preferences and outperform the traditional matrix factorization. Latent Relational Metric Learning (LRML) [40] extends CML by adding a relation vector. Pervious work [18, 19] apply metric learning loss to product search. All the metric learning algorithms are built on a static scenario and only learn a single representation for each entity. To the best of our knowledge, our DBML is the first metric learning algorithm that can simultaneously capture dynamics and uncertainty.

2.3 Dynamic Embedding Models

Our DBML model is a dynamic Bayesian model. A number of dynamic embedding [26–28, 39] models have appeared in the literature. To track the evolutions of users and items, dynamic latent space models have been proposed, e.g., collaborative Kalman fitering [39] and dynamic Poisson factorization [7]. Dynamic word embeddings [2] focus on tracking the evolutions of the semantics of words over time. To address the problem of dynamic user profiling, Liang et al. propose DUWE [28], an embedding model to jointly track the evolutions of users and words over time. Deng et al. propose LSM-RN [10], a dynamic latent space model to estimate how traffic patterns evolve. More recently, Chen and Li present STEP [8] to integrate both structural and temporal information in link prediction in dynamic networks. However, these dynamic embedding methods can not be directly applied to our product search task, in which there are three categories of entities, i.e., users, products and words, needed to be jointly inferred. Additionally, there are few related work to study the problem of dynamically modeling entities in a metric space.

3 NOTATIONS AND PROBLEM DEFINITION

Before we detail our DBML, we first formulate the dynamic personalized product search problem: let $\mathcal{U} = \{u_k\}_{k=1}^{N}$ and $\mathcal{I} = \{i_k\}_{k=1}^{M}$ be the sets of users and products, with N and M being the sizes of the sets, respectively. Let $\mathcal{D}^u_{\leq t} = \{\mathcal{D}^u_k\}_{k=1}^{t}$ be the streaming reviews provided by user u up to time t. Similarly, $\mathcal{D}^i_{\leq t} = \{\mathcal{D}^i_k\}_{k=1}^{t}$ denotes the streaming reviews provided for product i up to time

t. Each review document $\mathcal{D}$ ($\mathcal{D}_k^u$ or $\mathcal{D}_k^i$) contains a set of words: $\mathcal{D} = \{w_d\}_{d=1}^{|\mathcal{D}|}$, where w_d is from a vocabulary $\mathcal{V}$ of fixed size V and $|\mathcal{D}|$ is the number of words in $\mathcal{D}$. Let $\mathcal{S}_{\leq t} = \{\mathcal{S}_k\}_{k=1}^{t}$ be the streaming observed search records of user-query-product up to time t for all users $\mathcal{U}$, queries Q and products (items) $\mathcal{I}$, where each feedback in $\mathcal{S}_k$ is denoted as a tuple $s = (u, q, i)$.

Given a user u, his/her input query q, the streaming observed data $\mathcal{D}_{\leq t}^u$, $\mathcal{D}_{\leq t}^i$ and $\mathcal{S}_{\leq t}$, the goal of dynamic personalized product search is to predict the feedbacks $\mathcal{S}_t$ that meets her/his queries at every time t. Our DBML model is essentially a function f that seeks to find the following conditional probability indicating if user u prefers product i to i' at t given the input query q:

$$u, q, \mathcal{I}, \mathcal{D}_{\leq t}^{\mathcal{U}}, \mathcal{D}_{\leq t}^{\mathcal{I}} \mathcal{S}_{\leq t} \xrightarrow{f} p(i > i' | u, q, t, \mathcal{D}_{\leq t}^{\mathcal{U}}, \mathcal{D}_{\leq t}^{\mathcal{I}}, \mathcal{S}_{\leq t}), \quad (1)$$

where $\mathcal{D}_{\leq t}^{\mathcal{U}} = \{\mathcal{D}_{\leq t}^u\}_{u=1}^{|\mathcal{D}|}$ and $\mathcal{D}_{\leq t}^{\mathcal{I}} = \{\mathcal{D}_{\leq t}^i\}_{i=1}^{|\mathcal{I}|}$.

4 DYNAMIC BAYESIAN METRIC LEARNING

In this section, we first briefly review the latent vector model [1, 41] which extends the idea of skip-gram framework [32] to learn the entity representations and its limitations in §4.1. We then describe our dynamic Bayesian framework in §4.2.

4.1 Latent Vector Space Model

Recently, latent vector space model [1, 41] also called 'neural' representation learning [32] has been widely used in entity search. Latent vector space model learns the representations of entities based on the conditional probabilities of them over the observations in their contexts. Specifically, latent vector model maximizes the following log-likelihood of each context word w given the input entity e (which can be the product i or user u in our case):

$$\log p(\mathcal{D}|\mathcal{E}; \theta) = \sum_{e \in \mathcal{E}} \sum_{w \in \mathcal{D}^e} \log p(w|e), \quad (2)$$

where the probability $p(w|e)$ is parameterized as a softmax function $p(w|e) = \frac{\exp(\mathbf{e} \cdot \mathbf{w})}{\sum_{w \in \mathcal{D}^e} \exp(\mathbf{e} \cdot \mathbf{w})}$, where $\mathbf{w}$ and $\mathbf{e}$ are the latent representations of words and entities parameterized by θ. In product search [1, 41], the context $\mathcal{D}^e$ is the review documents for entity e. By applying negative sampling [32], Eq. 2 can be approximated as:

$$\log \sigma(\mathbf{w} \cdot \mathbf{e}) + \sum_{i=1}^{k} \mathbb{E}_{w_i' \sim p(w')} \log \sigma(-\mathbf{w}_i' \cdot \mathbf{e}), \quad (3)$$

where $p(w')$ is a pre-defined negative sampling distribution, k is the negative sample size and $\sigma(\mathbf{w} \cdot \mathbf{e}) = \frac{1}{1+\exp(-(\mathbf{w} \cdot \mathbf{e}))}$ is a sigmoid function. Although the latent vector space model has been achieved promising performance in entity search, the inner product score $s(w, e) = \mathbf{w} \cdot \mathbf{e}$ serving for the ranking causes some issues:

Contradicts. In the current latent vector space model, the score function $s(w, e) = ||\mathbf{w}|| \cdot ||\mathbf{e}|| \cdot \cos \alpha$ indicates those words or entities with large norms tend to have a larger score (even with small cosine score). So if we maximize Eq. 3, the most popular products may have large norms, however in real world, the most popular products are around the center of the entire products. In addition, in testing, since we want to keep the most similar entity is itself, we just use the cosine similarity $\frac{s(w,e)}{||\mathbf{w}|| \cdot ||\mathbf{e}||}$ to rank, which is also inconsistent with the $s(w, e)$ in training.

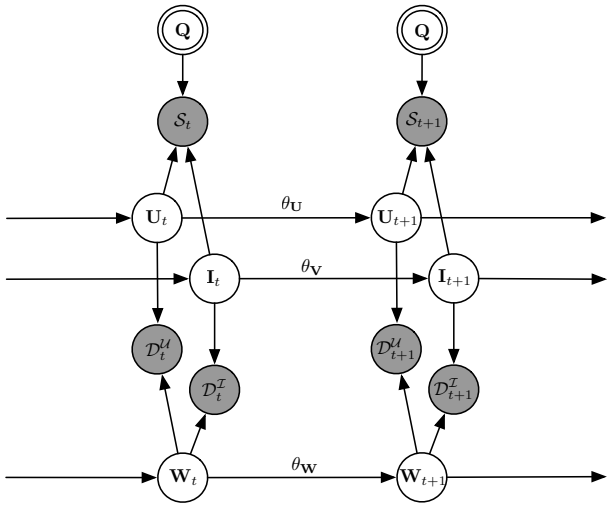

Figure 1: Graphical model of DBML. Gray, white and double circles represent observed, latent variables and parameters, respectively. Symbols above the lines denote parameters of the transition neural networks.

Triangle inequality. As discussed in previous work [21], the inner product does not satisfy the triangle inequality, which limits it to capture the fine grained preferences and the product-to-product or user-to-user similarity.

Implicit feedbacks. Implicit feedbacks (e.g., views or purchases) is available for product search. However, current latent vector models [1, 41] assume these unobserved feedbacks are negative, which is not reasonable as we do not know the fact that these feedbacks did not happen was because the user did not like the product or the user was not aware of it [21, 35].

Although some recent work such as CML [21] and LRML [40] which utilize metric learning idea to solve the problems above and may be applied to the our product search task, the assumptions of them are not realistic when it comes to the streaming scenario, where the latent semantic representations of words and entities change over time. In what follows, we detail our DBML model which can explicitly track the evolution of latent representations of words and entities in a metric space.

4.2 The Dynamic Bayesian Framework

To introduce our dynamic bayesian framework, we first model the likelihoods of the observed variables as follows:

Modeling the observed reviews. To address the contradicts, triangle inequality and implicit feedback problems of current latent vector model [1, 41] as mentioned in §4.1, we define the likelihood of the observed review provided by user u at time t as follows:

$$p(\mathcal{D}_t^{\mathcal{U}} | \mathbf{U}_t, \mathbf{W}_t) = \prod_{u \in \mathcal{U}} p(\mathcal{D}_t^u | \mathbf{u}_t, \mathbf{W}_t) = \prod_{u \in \mathcal{U}} \prod_{(w, w') \in \mathcal{T}_t^u} \sigma(\|\mathbf{u}_t - \mathbf{w}_t'\|_2^2 - \|\mathbf{u}_t - \mathbf{w}_t\|_2^2), \quad (4)$$

where $\mathcal{T}_t^u$ denotes the set of observed triplets from the observed reviews $\mathcal{D}_t^u$: $\mathcal{T}_t^u = \{(w, w') | w \in \mathcal{D}_t^u, w' \in \mathcal{V} \setminus \mathcal{D}_t^u\}$, $\sigma(x) = \frac{1}{1+\exp(-x)}$. Note that we don't add a constant threshold like pervious

metric learning methods [18, 21], since, in our experiments, we found our model is not sensitive to it . The motivation of the pariwise likelihood (4) is widely used in Bayesian personalized ranking (BPR) [35]. Different from the inner-product score used in BPR, we consider the euclidean distance score which can avoid contradicts and keep triangle inequality. For observed reviews $\mathcal{D}_t^{\mathcal{I}}$ provided by product i, we consider it is independently to $\mathcal{D}_t^{\mathcal{U}}$ [1] and define the following similar likelihood:

$$p(\mathcal{D}_t^{\mathcal{I}}|\mathbf{I}_t, \mathbf{W}_t) = \prod_{i \in \mathcal{I}} p(\mathcal{D}_t^i|\mathbf{i}_t, \mathbf{W}_t) = \prod_{i \in \mathcal{I}} \prod_{(w,w') \in \mathcal{T}_t^i} \sigma(\|\mathbf{i}_t - \mathbf{w}'_t\|_2^2 - \|\mathbf{i}_t - \mathbf{w}_t\|_2^2), \quad (5)$$

where $\mathcal{T}_t^i = \{(w, w')|w \in \mathcal{D}_t^i, w' \in \mathcal{V} \setminus \mathcal{D}_t^i\}$.

Modeling the observed feedbacks. The observed feedbacks in our task are triplets (user, query, product). For the query embedding $\mathbf{q}$, as discussed in [1], we can't learn it in an off-line way. Thus we use the query word embeddings to construct the query embedding:

$$\mathbf{q} = \tanh(\mathbf{A} \cdot \frac{\sum_{w_q \in q} \mathbf{w}_q}{|q|} + \mathbf{b}), \quad (6)$$

where $\mathbf{A}$ and $\mathbf{b}$ are the parameters of the tanh function, and $|q|$ is the number of words in the query. Note that one can use a more complexity function to obtain the query embeddings as in [1]. To be focused, we just apply Eq. 6 to obtain the embeddings, as it is simple yet effective as shown in the experiments. Note that since most words $\mathbf{w}_q$ in query (Eq. 6) are about category information for products [37, 41], we don't consider them as hidden variables like streaming review words but parameters, and don't model their evolutions for simplicity. Pervious work [1] utilizes $(\mathbf{u}+\mathbf{q})\cdot\mathbf{i}$ to model their relationships, which still suffers from contradicts problem and violates triangle inequality. In addition, it can not discover *analogical relations* [32]: given observed (u_1, q, i_1) and (u_2, q, i_2), we have $\mathbf{u}_1 - \mathbf{i}_1 + \mathbf{i}_2 \approx \mathbf{u}_2$. Thus, we define the likelihood of the observed feedbacks as:

$$p(\mathcal{S}_t|\mathbf{U}_t, \mathbf{I}_t; \mathbf{Q}) = \prod_{u \in \mathcal{U}} \prod_{(u,q,i,i') \in \mathcal{T}_t^s} \sigma(\|\mathbf{u}_t + \mathbf{q} - \mathbf{i}'_t\|_2^2 - \|\mathbf{u}_t + \mathbf{q} - \mathbf{i}_t\|_2^2), \quad (7)$$

where $\mathcal{T}_t^s = \{(u, q, i, i')|(u, q, i) \in \mathcal{S}_t, (u, q, i') \notin \mathcal{S}_t\}$ and $\mathbf{Q}$ is the set of all query embeddings $\mathbf{q}$. For brief discussion later, we let $p(\mathcal{S}_t|\mathbf{U}_t, \mathbf{I}_t)$ denote $p(\mathcal{S}_t|\mathbf{U}_t, \mathbf{I}_t; \mathbf{Q})$ whenever necessary. This formulation is inspired from knowledge graph embedding [6, 41], where two entities with should be close to each other under a certain relation operation. With the above likelihood design (Eq. 4, 5 and 7), our model can capture the affinities between review words, users and products in a unified metric space.

Modeling embeddings over time. To collaboratively track the evolutions of latent semantic representations of entities (i.e., users, products and words) over time, we need to define dynamic priors (transition distributions) for them. To model the evolution of the embeddings, such as the evolution of users' embeddings, $p(\mathbf{U}_t|\mathbf{U}_{t-1})$, pervious dynamic embedding methods [2, 28] consider Kalman filter as transition distribution:

$$p(\mathbf{U}_t|\mathbf{U}_{t-1}) \propto \mathcal{N}(\mathbf{U}_{t-1}, \boldsymbol{\sigma}_{t-1}^2)\mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_0^2), \quad (8)$$

where $\mathcal{N}(\mathbf{U}_{t-1}, \boldsymbol{\sigma}_{t-1}^2)$ and $\mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}_0^2)$ are the Gaussian distributions at time $t-1$ and 0, respectively, and $\boldsymbol{\sigma}_{t-1}^2$ and $\boldsymbol{\sigma}_0^2$ are the covariances of the Gaussian distributions at time $t-1$ and 0, respectively. The transition distribution is modeled as linear functions perturbed by Gaussian noise. However, in real world, the use of linear transition distribution limits the capacity to model complex phenomena. Thus we consider the following non-linear transition distribution as:

$$p(\mathbf{U}_t|\mathbf{U}_{t-1}) = \mathcal{N}(f(\mathbf{U}_{t-1}; \theta_{\mu_U}), \text{diag}(\exp(g(\mathbf{U}_{t-1}; \theta_{\sigma_U})))), \quad (9)$$

where f and g are non-linear functions parameterized by deep neural networks and $p(\mathbf{U}_1|\mathbf{U}_0) = \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. The transition distributions of words and products, i.e., $p(\mathbf{W}_t|\mathbf{W}_{t-1})$ and $p(\mathbf{I}_t|\mathbf{I}_{t-1})$, are defined similarly, thus we omit them. We use θ to denote all parameters of the neural networks corresponding to their transition distributions.

The unified model. Combining the designed likelihoods (Eq. 4, 5 and 7) and priors (Eq. 9), the joint distribution at time t of our model can be factorized as follows:

$$p(O_{1:T}, \mathcal{Z}_{1:T}; \mathbf{Q}, \theta) = \prod_{t=1}^{T} p(O_t|\mathcal{Z}_t; \mathbf{Q})p(\mathcal{Z}_t|\mathcal{Z}_{t-1}; \theta), \quad (10)$$

where $O_t = \{\mathcal{S}_t, \mathcal{D}_t^{\mathcal{U}}, \mathcal{D}_t^{\mathcal{I}}\}$ and $\mathcal{Z}_t = \{\mathbf{U}_t, \mathbf{I}_t, \mathbf{W}_t\}$ are the sets of observed and hidden variables at time t for brevity, respectively; and $p(O_t|\mathcal{Z}_t; \mathbf{Q})$ and $p(\mathcal{Z}_t|\mathcal{Z}_{t-1}; \theta)$ can be computed as:

$$p(O_t|\mathcal{Z}_t; \mathbf{Q}) = p(\mathcal{S}_t|\mathbf{U}_t, \mathbf{I}_t; \mathbf{Q})p(\mathcal{D}_t^{\mathcal{U}}|\mathbf{U}_t, \mathbf{W}_t)p(\mathcal{D}_t^{\mathcal{I}}|\mathbf{I}_t, \mathbf{W}_t), \quad (11)$$

$$p(\mathcal{Z}_t|\mathcal{Z}_{t-1}; \theta) = p(\mathbf{U}_t|\mathbf{U}_{t-1}; \theta)p(\mathbf{I}_t|\mathbf{I}_{t-1}; \theta)p(\mathbf{W}_t|\mathbf{W}_{t-1}; \theta), \quad (12)$$

Fig. 1 shows the graphical model of DBML. With this design, our model can model the affinities between entities and collaboratively track the non-linear evolutions of latent representations of them.

4.3 Online Inference

Given the joint distribution (Eq. 10), we need to infer the posteriors of the hidden variables ($\mathcal{Z}_t$) under a streaming scheme. Formally, we need to infer the following posterior one time step at a time:

$$p(\mathcal{Z}_t|O_{\leq t}) = p(O_t|\mathcal{Z}_t, O_{<t})p(\mathcal{Z}_t|O_{<t})/p(O_t|O_{<t}), \quad (13)$$

where, again, the parameters θ and $\mathbf{Q}$ are omitted for brevity. It is intractable to directly compute the posterior due to the nonlinear and non-conjugate likelihoods. As the review and feedback data arrive sequentially, we are interested in the posterior distribution over hidden variables conditioned on past but not on future observations: i.e., $p(\mathcal{Z}_t|O_{\leq t})$. We propose an online inference algorithm.

Variational Inference. We first rewrite the observation likelihood at time t as:

$$p(O_t|O_{<t}) = \int p(\mathcal{Z}_t|O_{<t})p(O_t|\mathcal{Z}_t)d\mathcal{Z}_t. \quad (14)$$

Note that $p(O_t|\mathcal{Z}_t) = p(O_t|\mathcal{Z}_t, O_{<t})$ due to the Markov assumptions underlying our model. Following variational inference [4], we can derive the following filtering Evidence Lower BOund (ELBO) of the log observation likelihood at time t as follows:

$$\log p(O_t|O_{<t}) = \mathbb{E}_{q(\mathcal{Z}_t)}\Big[\log \frac{p(O_t|\mathcal{Z}_t)p(\mathcal{Z}_t|O_{<t})q(\mathcal{Z}_t)}{p(\mathcal{Z}_t|O_{\leq t})q(\mathcal{Z}_t)}\Big] = \mathbb{E}_{q(\mathcal{Z}_t)}\big[\log p(O_t|\mathcal{Z}_t)\big] - \text{KL}\big(q(\mathcal{Z}_t)||p(\mathcal{Z}_t|O_{<t})\big) + \text{KL}\big(q(\mathcal{Z}_t)||p(\mathcal{Z}_t||O_{\leq t})\big) \quad (15)$$

$$
\begin{aligned}
&\geq \mathbb{E}_{q(\mathcal{Z}_t)}\big[\log p(O_t|\mathcal{Z}_t)\big] - \mathrm{KL}\big(q(\mathcal{Z}_t)||p(\mathcal{Z}_t|O_{<t})\big)\big] \\
&= \mathbb{E}_{q(\mathcal{Z}_t)}\big[\log p(O_t|\mathcal{Z}_t)\big] - \mathbb{E}_{q(\mathcal{Z}_t)}\big[\log q(\mathcal{Z}_t)\big] + \\
&\quad \mathbb{E}_{q(\mathcal{Z}_t)}\big[\log \mathbb{E}_{p(\mathcal{Z}_{t-1}|O_{<t})}[p(\mathcal{Z}_t|\mathcal{Z}_{t-1})]\big] \\
&= \mathbb{E}_{q(\mathcal{Z}_t)}\big[\log p(O_t|\mathcal{Z}_t)\big] - \mathbb{E}_{q(\mathcal{Z}_t)}\big[\log q(\mathcal{Z}_t)\big] + \\
&\quad \mathbb{E}_{q(\mathcal{Z}_t)}\big[\mathbb{E}_{p(\mathcal{Z}_{t-1}|O_{<t})}[\log p(\mathcal{Z}_t|\mathcal{Z}_{t-1})]\big] + \\
&\quad \mathbb{E}_{q(\mathcal{Z}_t)}\big[\mathrm{KL}\big(p(\mathcal{Z}_{t-1}|O_{<t})||p(\mathcal{Z}_{t-1}|\mathcal{Z}_t, O_{<t})\big)\big] \\
&\geq \mathbb{E}_{q(\mathcal{Z}_t)}\big[\log p(O_t|\mathcal{Z}_t)\big] - \mathbb{E}_{q(\mathcal{Z}_t)}\big[\log q(\mathcal{Z}_t)\big] + \\
&\quad \mathbb{E}_{q(\mathcal{Z}_t)}\big[\mathbb{E}_{p(\mathcal{Z}_{t-1}|O_{<t})}[\log p(\mathcal{Z}_t|\mathcal{Z}_{t-1})]\big] = \mathcal{L}_t, \qquad (16)
\end{aligned}
$$

where $\mathrm{KL}(\cdot||\cdot)$ is the KL divergence, the parameters are omitted for brevity and $q(\mathcal{Z}_t)$ is the introduced mean-field variational distribution corresponding the hidden variables:

$$
q(\mathcal{Z}_t;\phi) = \prod_{u\in\mathcal{U}}\prod_{i\in\mathcal{I}}\prod_{w\in\mathcal{V}} q(\mathbf{u}_t;\phi_u)q(\mathbf{i}_t;\phi_i)q(\mathbf{w}_t;\phi_w). \qquad (17)
$$

Amortization. To obtain more flexible variational distribution, unlike to pervious works [2, 28] in which they directly introduce parameters for every variational distribution at every time, we consider amortized inference [16, 23] to avoid to optimize parameter for each local variational distribution; instead, it fits a non-linear neural network to calculate each variational parameter. For instance, the variational distribution $q(\mathbf{u}_t;\phi_u)$ is defined as:

$$
q(\mathbf{u}_t|\mathbf{x}_u;\phi_u) = \mathcal{N}(\mu_{\phi_U}(\mathbf{x}_u), \mathrm{diag}(\exp(\Sigma_{\phi_U}(\mathbf{x}_u)))), \qquad (18)
$$

where $\mu_{\phi_U}(\mathbf{x}_u)$ and $\Sigma_{\phi_U}(\mathbf{x}_u)$ are two outputs of a feed-forward non-linear neural networks parametrized by ϕ_U like vanilla variational auto-encoders [23]. $\mathbf{x}_u$ is the attribute vector of user u. Note that ϕ_U is shared across all users and all time slices thus enjoy statistical strength benefits. $q(\mathbf{i}_t|\mathbf{x}_i;\phi_I)$ and $q(\mathbf{w}_t|\mathbf{x}_w;\phi_W)$ are similarly defined and conditioned their on own attributes, thus we omit them here. We use $\mathcal{X}$ to denote the set of attributes of all entities and define $\phi = \{\phi_U, \phi_I, \phi_W\}$.

Recursive update. Maximizing the lower bound (Eq. 16) is equal to minimize the Kullback-Leibler (KL) divergence between variational distribution $q(\mathcal{Z}_t|\mathcal{X};\phi)$ and real posterior $p(\mathcal{Z}_t|O_{\leq t})$. The expectation term in Eq. 16 can be estimated with low variance using reparameterization trick [23]. However, the posterior $p(\mathcal{Z}_t|O_{<t})$, is still intractable. We recursively estimate it as follows using the inferred prior $q(\mathcal{Z}_{t-1})$ at previous time slice:

$$
\begin{aligned}
\mathcal{L}_t &= \mathbb{E}_{q(\mathcal{Z}_t|\mathcal{X};\phi)}\big[\log p(O_t|\mathcal{Z}_t;\mathbf{Q})\big] - \qquad (19)\\
&\quad \mathbb{E}_{q(\mathcal{Z}_t|\mathcal{X};\phi)}\big[\log q(\mathcal{Z}_t|\mathcal{X};\phi) - \mathbb{E}_{q(\mathcal{Z}_{t-1})}[\log p(\mathcal{Z}_t|\mathcal{Z}_{t-1};\theta)]\big],
\end{aligned}
$$

Combining Eq. 19 and 16, and applying reparameterization trick [23], the lower bound $\mathcal{L}_t$ can approximated as follows:

$$
\mathcal{L}_t = \log p(O_t|\hat{\mathcal{Z}}_t;\mathbf{Q}) - \mathbb{E}_{q(\mathcal{Z}_t|\mathcal{X};\phi)}\big[\log\big(\frac{q(\mathcal{Z}_t|\mathcal{X};\phi)}{p(\mathcal{Z}_t|\hat{\mathcal{Z}}_{t-1};\theta)}\big)\big]. \qquad (20)
$$

where $\hat{\mathcal{Z}}_t$ and $\hat{\mathcal{Z}}_{t-1}$ are randomly drawn from the current posterior $q(\mathcal{Z}_t|\mathcal{X};\phi)$ and the already inferred posterior $q(\mathcal{Z}_{t-1})$, respectively. The expectation term is about two Gaussians, thus it have a closed form. To perform online inference and parameter estimation, We can present a variational expectation maximization (EM) algorithm. In the **E** step, we take a gradient step with respect to the variational parameters ϕ to find $q(\mathcal{Z}_t|\mathcal{X};\phi)$ that tightens the lower bound $\mathcal{L}_t(\phi,\theta,\mathbf{Q})$. This can be achieve by setting $q(\mathcal{Z}_t|\mathcal{X};\phi)$ to $p(\mathcal{Z}_t|O_{\leq t})$, since the KL divergence will minimize. In the **M** step, we take a gradient step with respect to the model parameters (θ and $\mathbf{Q}$) to maximize the lower bound $\mathcal{L}_t(\phi,\theta,\mathbf{Q})$, with $q(\mathcal{Z}_t|\mathcal{X};\phi)$ fixed to the value found in the E-step. we can iteratively conduct E-step and M-step until convergence. Like discussed in pervious works [23, 24], this method requires potentially an expensive iterative optimization, thus we directly update θ, ϕ and $\mathbf{Q}$ jointly. The overview of online inference is shown in Algorithm 1.

Algorithm 1: Online Inference at t.

Input: Observed reviews $\mathcal{T}_t^{\mathcal{U}}$, $\mathcal{T}_t^{\mathcal{I}}$ and feedback $\mathcal{T}_t^{\mathcal{S}}$. Inferred posterior at last time, i.e., $q(\mathcal{Z}_{t-1}|\mathcal{X};\phi)$;
Output: The posterior $q(\mathcal{Z}_t|\mathcal{X};\phi)$ at current time t;

```
q(Z_{t-1}) = q(Z_{t-1}|X;φ);
while not convergent do
    Randomly draw the mini-batch training set O_t^M from the constructed observed T_t^U, T_t^I and T_t^S;
    Draw Ẑ_t and Ẑ_{t-1} from q(Z_t|X;φ) and q(Z_{t-1}) with the reparametrization trick;
    Update parameters using ∇_{θ,φ} L_t with mini-batch O_t^M;
end
Return θ, φ and the posterior at current time q(Z_t|X;φ).
```

Time complexity. The time complexity of the training procedure is $O(NB(K+C))$, where N is the number of iterations and B represents the batch size, K is the latent dimension and C represents the wights of three transition neural networks respect to users, products and words. Typically, each transition neural network is just two-layer. The algorithm can be efficiently accelerated using multi-core CPUs or GPUs with our Pytorch implementation.

4.4 Offline Inference

Although our online inference algorithm can sequentially and adaptively learn the latent embeddings from the streaming observation and is very efficient, it infers $q(\mathcal{Z}_t)$ only conditioned on past observations $O_{\leq t}$ until a given time t. However, for offline situation in real world, we have the entire sequence of observation. Thus we propose an offline algorithm to infer $q(\mathcal{Z}_{1:T})$ given the sequence of observation $O_{1:T}$:

$$
p(\mathcal{Z}_{1:T}|O_{1:T}) = \frac{p(\mathcal{Z}_{1:T}, O_{1:T})}{\int p(\mathcal{Z}_{1:T}, O_{1:T}) d\mathcal{Z}_{1:T}}, \qquad (21)
$$

where T is the end of observed time slices, and we omit the parameters for brevity. The problem is that the normalization is intractable. Similar to the proposed online inference, we also adopt the idea of variational inference [4] to learn $p(\mathcal{Z}_{1:T}|O_{1:T})$ in a offline way.

The evidence lower bound. Following variational inference [4], we could derive the variational evidence lower bound corresponding to the log-marginal likelihood of observed sequences across all time slices as follows:

$$
\begin{aligned}
\log p(O_{1:T}) &= \log \int p(\mathcal{Z}_{1:T}, O_{1:T}) d\mathcal{Z}_{1:T} \\
&= \log \int \prod_{t=1}^{T} q(\mathcal{Z}_t) \prod_{t=1}^{T} \frac{p(\mathcal{Z}_t|\mathcal{Z}_{t-1}) p(O_t|\mathcal{Z}_t)}{q(\mathcal{Z}_t)} d\mathcal{Z}_{1:T}
\end{aligned}
$$

$$\geq \int \prod_{t=1}^{T} q(\mathcal{Z}_t) \sum_{t=1}^{T} \log \frac{p(\mathcal{Z}_t|\mathcal{Z}_{t-1})p(O_t|\mathcal{Z}_t)}{q(\mathcal{Z}_t)} d\mathcal{Z}_{1:T}$$
$$= \sum_{t=1}^{T} \mathbb{E}_{q(\mathcal{Z}_t)}\big[\log p(O_t|\mathcal{Z}_t)\big] - \mathrm{KL}(q(\mathcal{Z}_1)||p(\mathcal{Z}_1|\mathcal{Z}_0))$$
$$- \sum_{t=2}^{T} \mathbb{E}_{q(\mathcal{Z}_{t-1})}\big[\mathrm{KL}(q(\mathcal{Z}_t)||p(\mathcal{Z}_t|\mathcal{Z}_{t-1})\big] = \mathcal{L}. \quad (22)$$

where $q(\mathcal{Z}_t)$ is the introduced variational distribution as same as in online inference. Unlike online inference, We can estimate ELBO by sampling mini-batch in the arbitrary two adjacent time slices t and $t+1$:

$$\mathcal{L}_t = \mathbb{E}_{q(\mathcal{Z}_t|\mathcal{X},t;\phi)}\big[\log p(O_t|\mathcal{Z}_t;\mathbf{Q})\big] - \mathbb{E}_{q(\mathcal{Z}_t|\mathcal{X},t;\phi)}\big[\quad (23)$$
$$\log q(\mathcal{Z}_t|\mathcal{X},t;\phi) - \mathbb{E}_{q(\mathcal{Z}_{t-1}|\mathcal{X},t-1)}[\log p(\mathcal{Z}_t|\mathcal{Z}_{t-1};\theta)]\big],$$

where $\mathbf{x}_u$ represents the attribute vector of u, t is a scalar timestep, ϕ_U represents variational parameters. $\mu_{\phi_U}(\mathbf{x}_u)$ and $\Sigma_{\phi_U}(\mathbf{x}_u)$ are two outputs of a DiffTime network [36]. We can estimates it via Monte Carlo estimates and the reparameterization trick [23]. Maximizing the ELBO is equivalent to minimize the KL divergence between variational distribution $q(\mathcal{Z}_{1:T}|\mathcal{X},1:T;\phi)$ and $p(\mathcal{Z}_{1:T}|O_{1:T})$. Although offline inference algorithm can achieve higher likelihood, it need recompute the latent embeddings of all the entities from at every time stamp and record the computed variation distribution $q(\mathcal{Z}_t|\mathcal{X},t;\phi)$ as $q(\mathcal{Z}_t|\mathcal{X},t)$ for each time stamp t. The computation cost of offline inference of one iteration is as same as online inference, however it typically needs more iteration to converge (see §6.3 for details). Leveraging offline or online inference algorithms our DBML model can strike a balance between accuracy and efficiency.

4.5 Prediction

After the parameters $\mathbf{Q}$ and θ are estimated, and the $q(\mathcal{Z}_t)$ i.e., $q(\mathbf{U}_t)$, $q(\mathbf{I}_t)$ and $q(\mathbf{W}_t)$ are inferred, The probability that a user prefers product i to product i' given the query q at time t can be predicted by Bayesian point estimation as :

$$p(i > i'|u,q,t,\mathcal{D}^{\mathcal{U}}_{\leq t},\mathcal{D}^{\mathcal{I}}_{\leq t},\mathcal{S}_{\leq t}) =$$
$$q(\mathcal{Z}_t)\cdot\sigma(\|\mathbf{u}_t+\mathbf{q}-\mathbf{i}'_t\|_2^2 - \|\mathbf{u}_t+\mathbf{q}-\mathbf{i}_t\|_2^2)$$
$$= \sigma(\|\hat{\mathbf{u}}_t+\mathbf{q}-\hat{\mathbf{i}}'_t\|_2^2 - \|\hat{\mathbf{u}}_t+\mathbf{q}-\hat{\mathbf{i}}_t\|_2^2), \quad (24)$$

where $\hat{\mathbf{u}}_t$, $\hat{\mathbf{i}}'_t$ and $\hat{\mathbf{i}}_t$ are drawn from $q(\mathcal{Z}_t)$. The whole product rank list can be easily obtained by the above pair-wise probability.

5 EXPERIMENTAL SETUP

In this section, we detail our experimental setup.

5.1 Research Questions

In what follows, we seek to answer the research questions that guide the remainder of this paper: **(RQ1)** Can DBML achieve better performance for product search, compared to the state-of-the-art static and dynamic models? **(RQ2)** What is the effect of the embedding size on the performance of our DBML? **(RQ3)** What is impact of the number of iterations on the retrieval performance of our proposed DBML and its variants? **(RQ4)** Can the learned product embeddings distinguish different types of products and capture the uncertainties of them? **(RQ5)** Can the inferred dynamic embeddings help to capture the popularity of the products and users' interests over time?

Table 1: Statistics of the four datasets.

Datasets	*Electronics*	*Toys*	*Clothing*	*Cell Phones*
#reviews	1,689,188	167,597	675,929	194,439
#queries	989	407	2,041	165
#users	19,403	19,421	39,387	27,879
#products	63,001	11,924	23,033	10,429
Review per user	8.78	8.63	17.16	6.97
Review per product	26.81	14.06	29.35	18.64
Avg query length	6.40	6.01	6.39	5.93
Query per user	8.13	8.48	17.16	4.95
Query per product	1.02	1.07	2.33	1.11
Time windows	15	12	9	8
Training triples	1,421,050	125,862	597,155	172,590
Testing triples	192,403	19,403	39,387	27,879

5.2 Datasets

In order to answer our research questions, we evaluate our model and the baselines on the Amazon product dataset[1] provided by McAuley et al. [30]. The Amazon product dataset contains millions of users and products as well as reviews and product metadatas from May, 1996 to July, 2014. In our experiments, following pervious work [1, 18, 41], we extract four sub-datasets from the dataset: *Electronics*, *Toys*, *Clothing* and *Cell Phones*. We use the 5-core data [20, 30] in which all users and products have at least 5 reviews. Following the paradigm used in the pervious work [1, 18, 41], we extract the corresponding search query from the categories to which the product belongs. Please refer to work [1, 41] for the details of query extraction. Following [1, 18, 41], we construct user-query pairs by linking user-product pairs with each product's queries for personalized product search. Since we consider streaming scenario, where users' search and review comments arrive sequentially over time, we split each data set into time slices using the time stamps provided by the dataset. Without loss of generality, we let the duration of a time slice be one year for all datasets. Note that the duration can be selected manually to suit different datasets in real world. For each dataset, following [18], the last purchasing transaction of each user is held for the testing set Table 1 shows the statistics of the four datasets we used.

5.3 Baselines

We make comparisons between our DBML model [2] and the following state-of-the-art algorithms for personalized product search:
Traditional regression method:
Logistic Regression Search Model (LRS): This method [43] is the winner of CIKM Cup 2016 competition introduced by. Pervious work [18] adapted the second best model in the original paper [43] to the task of product search.
Bag-of-words representation models:
Query Likelihood Model (QL): This method [33] is a smoothing language modeling approach. Pervious work [1] utilizes it to estimate a language model for each product, and then ranks product

[1] http://jmcauley.ucsd.edu/data/amazon/
[2] Code are publicly available from https://github.com/DBML-model/DBML.

by the likelihood of the query according to the estimated language model.

Extended Query Likelihood with User Models (UQL): This model is an extended QL introduced by [1] , in which personalization is taken into account.

Latent vector space models:

Latent Semantic Entity (LSE): This method [41] is designed for product search task, in which words and products are embedded into the same latent space.

Hierarchical Embedding Model (HEM): This the state-of-the-art method [1], which extends LSE to a personalized search model by adding user models.

Dynamic product search models:

Attentive Long Short-Term Preference model (ALSTP) : This model [18] is the state-of-the-art approach for the dynamic personalized product search. It utilizes RNN and attention mechanism to capture long- and short-term user preferences over time.

We consider following variant versions of DBML:

DBSK: The original HEM model is not a dynamic product search model which is based on skip-gram loss function, so we extended it to a dynamic version by considering the evolution of users, words and products and using our proposed dynamic bayesian framework. We call this model as Dynamic Bayesian SKip-gram model (DBSK).

DBML-point: For comparisons, we revise the ranking likelihood (Eq. 4, 5 and 7) to the point-wise likelihood in metric space and we conduct online inference on it.. For instance, Eq. 4 is changed to (The same revisions are applied to Eq. 5 and Eq. 7): $p(\mathcal{T}_t^{\mathcal{U}}|\mathbf{u}_t, \mathbf{W}_t) = \prod_{u\in\mathcal{U}} \prod_{w\in\mathcal{D}_t^u} \sigma(\|\mathbf{u}_t - \mathbf{w}_t\|_2^2) \cdot \prod_{w\in\mathcal{V}-\mathcal{D}_t^u} \sigma(-\|\mathbf{u}_t - \mathbf{w}'_t\|_2^2)$.

DBML-online This Dynamic Bayesian Metric Learning model proposed in this paper conduct proposed online inference.

DBML-offline This Dynamic Bayesian Metric Learning model proposed in this paper conduct proposed offline inference.

5.4 Evaluation Metrics

To evaluate the product search performance, as in [1, 18, 41], we use a number of widely used evaluation metrics, i.e., HR@k (Hit Ratio at k) [18], MRR@k (Mean Reciprocal Rank at k) [9] and NDCG@k (Normalized Discounted Cumulative Gain at k) [1, 9]. We compute the scores at depth 20, i.e., let k =20. We report the average results of all test users.

5.5 Experimental Settings

For LRS and QL baselines, their parameters are set as the same as those in the pervious work [1, 18, 41]. For LSE [3], HEM [4] and ALSTP [5], we use the publicly available source codes provided by the authors to conduct our experiments and find the hyper-parameters of them via extensive grid search. For our DBSK, DBML-point, DBML-offline and DBML-online, we implement them by ourselves. All parameters are initialized using the uniform distribution in the range $[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}]$, where n is the number of the input of the model. The number of negative samplings for each positive training data at each batch is set to 5 for all methods. We use AdaGrad [13] with momentum to optimize our model and the baselines. The learning rate is set to be 0.01. The embedding sizes of our methods and the baselines are set to 50. We consider attribute vectors $\mathbf{x}_u$, $\mathbf{x}_i$ and $\mathbf{x}_w$ as their id's one-hot vectors. The effects of embedding size on the final performances are examined §6.2. All the models are trained from scratch without any pre-training on a single NVIDIA GeForce GTX 1060 GPU with a batch size of 256.

[3] https://github.com/cvangysel/SERT.
[4] https://ciir.cs.umass.edu/downloads/HEM/.
[5] https://github.com/guoyang9/ALSTP.

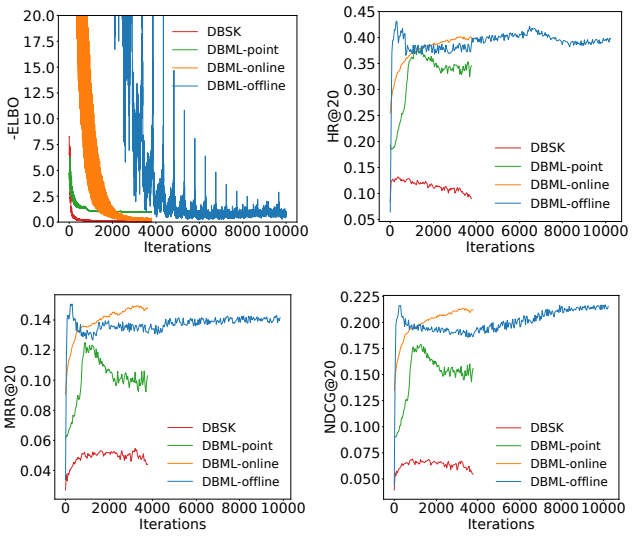

Figure 2: Negative ELBO, HR@20, MRR@20 and NDCG@20 performance on the *Toys* with the number of iterations from 0 to 10000, respectively.

6 RESULTS AND ANALYSIS

6.1 Overall Performance Comparison

RQ1: We compare the search performance of our DBML with that of the baselines listed in §5.3. Table 2 summarized the best results of all the baselines and our model on four benchmark datasets. We design four types of DBML, which are DBSK, DBML-point DBML-online and DBML-offline; see §5.3. From Table 2, we have the following findings: (1) Both our DBML-online and DBML-offline statistically significantly outperform all the baselines on all the metrics, which confirms the effectiveness of our dynamic Bayesian framework for the product search. (2) All the dynamic methods, i.e., DBSK, DBML-point, DBML-online, DBML-offline and ALSTP, outperform non-dynamic baselines such as HEM and LSE, which demonstrates that capturing the evolutions of entities helps to improve the product search performance. (3) Our DBML-offline outperforms DBML-online. This is expected since DBML-offline shares information across all the time (future and past) and uses a more flexible variational distribution. (4) In term of designed likelihood, our DBML-online outperforms DBML-point and DBSK, which demonstrates our pair-wise metric likelihood is more better than point-wise metric and skip-gram likelihoods. We attribute it to that our pair-wise metric likelihood can avoid contradicting, keep triangle inequality and suit to implicit feedbacks. (5) The dynamic version of HEM, i.e., DBSK, outperforms HEM. This again shows that our dynamic Bayesian does can capture the evolutions of entities and achieves better performance.

Table 2: Product search performance on the *Electronics*, *Toys*, *Clothing* and *Cell Phones* datasets in terms of HR@20 (HR), MRR@20 (MRR) and NDCG@20 (NDCG). The best and the second best performance of the methods per metric and per dataset are marked in boldface and at the upper corner of the scores by *, respectively. The best baselines per metric and per dataset are marked at the upper corner of the scores by †.

	Electronics			*Toys*			*Clothing*			*Cell Phones*		
	HR	MRR	NDCG	HR	MRR	NDCG	HR	MRR	NDCG	HR	MRR	NDCG
LRS	0.069	0.016	0.028	0.150	0.042	0.077	0.092	0.054	0.077	0.050	0.014	0.035
QL	0.080	0.039	0.034	0.105	0.041	0.049	0.086	0.047	0.063	0.058	0.015	0.032
UQL	0.110	0.043	0.051	0.112	0.052	0.054	0.093	0.053	0.075	0.058	0.018	0.027
LSE	0.150	0.062	0.062	0.115	0.043	0.056	0.158	0.067	0.108	0.064	0.025	0.032
HEM	0.189	0.076	0.083	0.131	0.054	0.072	0.186	0.075	0.126	0.076	0.036	0.033
ALSTP	$0.212^\dagger$	$0.094^\dagger$	$0.109^\dagger$	$0.302^\dagger$	$0.087^\dagger$	$0.123^\dagger$	$0.279^\dagger$	$0.097^\dagger$	$0.152^\dagger$	$0.194^\dagger$	$0.065^\dagger$	$0.096^\dagger$
DBSK	0.270	0.082	0.107	0.142	0.062	0.081	0.321	0.103	0.167	0.086	0.043	0.045
DBML-point	0.234	0.056	0.095	0.376	0.107	0.166	0.312	0.084	0.134	0.112	0.034	0.039
DBML-online	0.388^*	0.129^*	0.185^*	0.430^*	0.138^*	0.201^*	0.349^*	0.140^*	0.186^*	0.251^*	0.078^*	0.085^*
DBML-offline	**0.394**	**0.132**	**0.190**	**0.438**	**0.138**	**0.204**	**0.425**	**0.234**	**0.280**	**0.260**	**0.086**	**0.124**

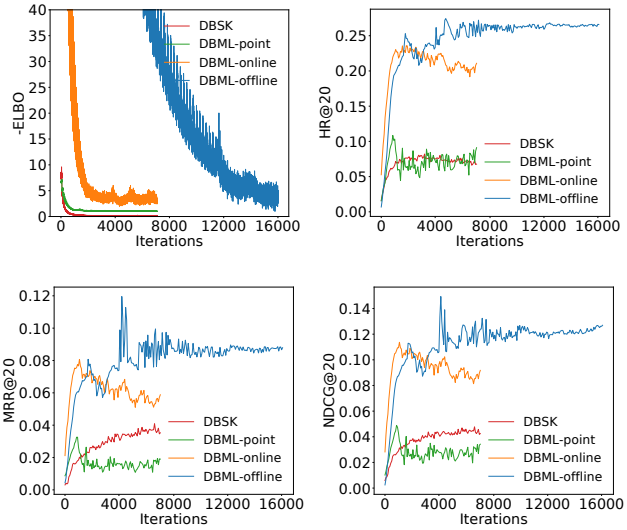

Figure 3: Negative ELBO, HR@20, MRR@20 and NDCG@20 performance on the *Cell Phones* dataset with the number of iterations from 0 to 16000, respectively.

6.2 Dimensions of Embeddings

RQ2: We now study how the dimension of the embedding affects the search performance. Fig. 4 shows the NDCG@20 performance of our proposed model and and the best baselines, ALSTP, HEM and LSE, on different sizes of dimensions varying between 25 and 100 on *Toy* and *Cell Phones*. Note that we don't report other metrics, as it follows the same pattern. According to the Fig. 4, we can find that DBML-offline significantly and consistently outperforms all other baselines and DBML-online at all different sizes of dimensions, which demonstrates that the effectiveness of our propose offline inference algorithm. That the variant DBSK outperforms the static HEM illustrates that dynamic embeddings inferred by our dynamic Bayesian framework can lead to better performance. When the size of dimension increases from a 75 to 100, DBML-offline reaches a plateau but still outperforms the best baselines. All of these findings demonstrate that DBML and its variants are robust and they are able to maintain significant improvements of product search performance over the state-of-the-art.

6.3 Numbers of Iterations

RQ3: To figure out the impact of the number of iterations on the product search performance, we plot the negative ELBO and performance curves of our DBSK, DBML-point, DBML-online and DBML-offline models with different number of iterations from 0 to convergence on datasets *Toys* and *Cell Phones* in terms of the representative evaluation metrics: loss (negative ELBO), HR@20, MRR@20 and NDCG@20 in Fig. 2 and 3. Note that HR@20, MRR@20 and NDCG@20 is evaluated on the test set. From Fig. 2 and 3, we can find: (1) Compared offline inference (i.e., DBML-offline) with online inference (i.e., DBSK, DBML-point, DBML-online), the offline inference needs more iterations to converge (achieves small negative ELBO), This is expected since offline inference algorithm based entire observation across all time slices from the past to the future. (2) The performance of all four models gradually increases with the iterations from 0 to 2,000. In addition, DBML-online consistently outperforms DBML-point and DBSK, and such performance gap seems to increase with more iterations. This again confirms that our pair-wise metic likelihood strategies are effective. (3) In general, DBML-offline outperforms DBML-online with the iterations increase, which demonstrates that offline inference can infer better latent embeddings and estimate more accurate model parameters. (4) All the four methods tend to overfit the observation with the their negative ELBOs decrease, indicating that early terminating the iterations early can achieve better performance.

6.4 Quality of Product Embeddings

RQ4: One of the most important properties of our model is that the learned product embeddings are interpretable and can distinguish different types of products. Therefore, we develop a test to see if products can be categorized by meaning based on embeddings. Due to the space limitations, we only extracted 10 categories of product embeddings learned by DBML-online on *Toys* and *Cell Phones* at the last time and map those embeddings into 2D with T-SNE [29]. Since our method is able to learn full posteriors of latent embeddings, following [5, 31], we also plot the variances of the

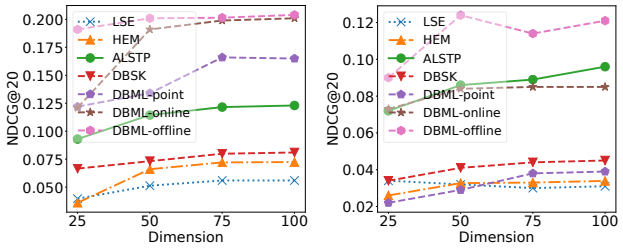

Figure 4: NDCG@20 performance with various sizes of dimensions of embeddings on datasets *Toys* and *Cell Phones*.

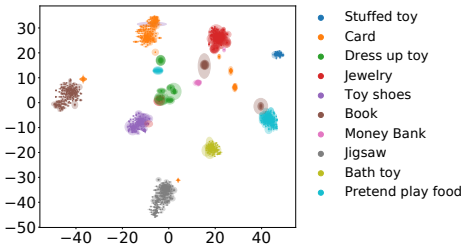

Figure 5: The product embeddings in the *Toys* dataset.

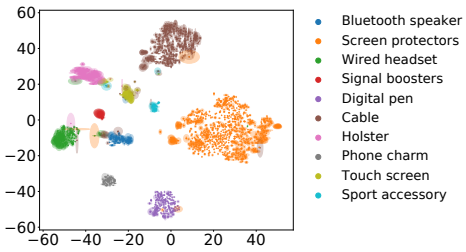

Figure 6: The product embeddings in the *Cell Phones* dataset.

resulting posteriors to visualize the products' uncertainties. Fig. 5 and Fig. 6 show the results. From Fig. 5 and Fig. 6, we can find that the different categories of products are clearly separated in both two datasets. We can also see that some of those products which are far from their cluster centers tend to have large variances, which illustrates that the posteriors learned by our model can help us to capture the positions of the products by the mean vectors and the uncertainties of products by the variances.

6.5 Evolution of Products and Users

RQ5: Finally, we examine whether DBML can capture the evolution of user preferences and product popularity over time. To show the evolution of product popularity, we consider the following configuration: we first compute the average of all inferred latent embeddings of users at each time $\tilde{\mathcal{U}} = \{\tilde{\mathbf{u}}_k\}_{k=1}^{T}$ as public preferences. Then, we randomly choose 10 brands of products and use the averages of those embeddings at each time as the embedding of the brand b at that time $\tilde{I}^b = \{\tilde{\mathbf{i}}_t^b\}_{t=1}^{T}$. We define the following probability as the popularity of brand b at t: $pop(b_t) = \frac{\exp(-||\tilde{\mathbf{u}}_k + \mathbf{q} - \tilde{\mathbf{i}}_t^b||)}{\sum_{b'=1}^{10} \exp(-||\tilde{\mathbf{u}}_t + \mathbf{q} - \tilde{\mathbf{i}}_t^{b'}||)}$, where $\mathbf{q}$ is query embedding of q that related to the dataset. E.g., the queries are *toys games* and *accessories cell phones* in *Toys* and *Cell Phones* datasets, respectively.

Table 3: Top-5 predicted products' categories of two example users' dynamic search results from *Cell Phones* (top subtable) and *Toys* (bottom subtable) over 2011 to 2014.

Years	Ground Truth	DCRL-online
2011	Chargers	Chargers, Car Chargers, Power Adapters, Vehicle Electronics, Kindle Store
2012	Basic Cases	Basic Cases, Wallet Cases, Waterproof Cases, Portable Audio, MP3
2013	Batteries, Chargers Cases, Basic Cases	Basic Cases, Chargers Cases, Electronics, Touch Screen, Tablet Adapters
2014	Watches, Clothing, Sports, Electronics	Watches, Computers , Clothing, Sports, Electronics
2011	Sports , Outdoor Play ,Blasters Foam	Sports, Outdoor Play, Blasters Foam, Balloons, Party Supplies
2012	Baby, Bath ,Puzzles, Dolls, Maze	Baby, Puzzles, Bath, Dolls ,Sequential
2013	Arts Craft, Kits ,Jewelry	Arts Craft, Jewelry, Craft Kids, Kits,Sewing
2014	Games, Card Games	Game, Card Games, Arts Craft, Part Supplies, Toys Games

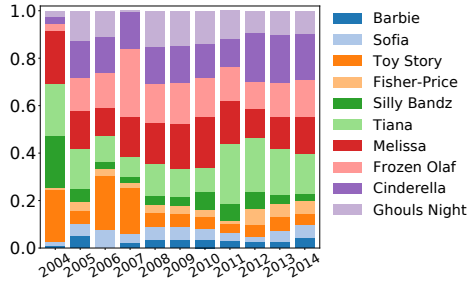

Figure 7: The evolution of brands in the *Toys* dataset.

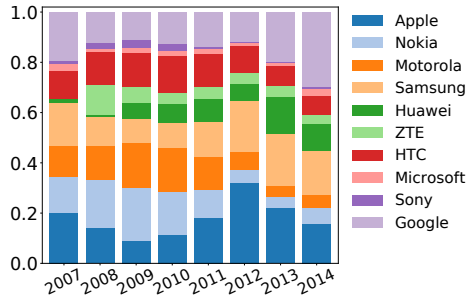

Figure 8: The evolution of brands in the *Cell Phones* dataset.

Fig. 7 shows that Toy Story became popular from 2005 to 2007. Over 250,000 figures were sold for each character before the film released, demands expanded, and eventually reached over 25 million units sold in 2007. [6] Fig. 8 shows that evolution of popularity of different kind of brands in dataset *Cell Phones*. From Fig. 8, We can notice that some brands have gained in popularity such as "Google" , "Huawei" and "Samsung" from 2009 to 2014. On the contrary, the "Nokia" and "Motorola" are out of date. These findings are consistent in our real world. All of these findings demonstrate that our model can better explain and capture the product popularity over time.

[6] https://en.wikipedia.org/wiki/Toy_Story.

To evaluate whether our DBML-online can capture the evolution of user preferences over time, we randomly choose two example users and show the searched product (ground truth) and the top-5 predicted by DBML-online at every time in the *Cell Phones* and *Toy* datasets. From Table 3, we can find first user's interest from basic cases move to watches and clothing, and the second user's interest from outdoor play move to art craft and card games. Our DBML-online can track the users' interests over time and return the top-5 relevant products' categories at different times, which demonstrates the effectiveness of our dynamic model and the high quality of the dynamic embeddings.

7 CONCLUSION

In this paper, we have studied the problem of personalized product search under a streaming scenario, where the search feedbacks and reviews arrive sequentially over time. To address this problem, we have proposed a dynamic Bayesian framework to explicitly and jointly track the evolutions of users, products and words in a metric space. The propose pari-wise metric likelihood makes our DBML able to avoid contradicting and keep triangle inequality in the training. The proposed non-linear priors lead to capturing the evolutions of entities. To infer the dynamic embeddings, we have proposed two scalable and effective variational inference algorithms, based on amortized inference, i.e., the online and offline inference algorithms to balance between accuracy and efficiency. Experimental results show our DBML can capture the evolutions of different entities and achieve better performance. Results also show our model does capture the evolutions and uncertainties of entities by the inferred latent embeddings and the latent embeddings are highly interpretable to explain the real world.

Acknowledgments. This research was partially supported by the National Natural Science Foundation of China (Grant No. 61906219).

REFERENCES

[1] Qingyao Ai, Yongfeng Zhang, Keping Bi, Xu Chen, and W Bruce Croft. 2017. Learning a hierarchical embedding model for personalized product search. In *SIGIR*. ACM, 645–654.
[2] Robert Bamler and Stephan Mandt. 2017. Dynamic Word Embeddings. In *ICML*. 380–389.
[3] Aurélien Bellet, Amaury Habrard, and Marc Sebban. 2013. A Survey on Metric Learning for Feature Vectors and Structured Data. *CoRR* abs/1306.6709 (2013).
[4] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2017. Variational inference: A review for statisticians. *J. Amer. Statist. Assoc.* (2017), 859–877.
[5] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *ICLR*.
[6] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*. 2787–2795.
[7] Laurent Charlin, Rajesh Ranganath, James McInerney, and David M. Blei. 2015. Dynamic Poisson Factorization. In *ACM RecSys*. 155–162.
[8] Huiyuan Chen and Jing Li. 2018. Exploiting Structural and Temporal Evolution in Dynamic Link Prediction. In *CIKM*. 427–436.
[9] Bruce Croft, Donald Metzler, and Trevor Strohman. 2009. Search Engines: Information Retrieval in Practice. (2009).
[10] Dingxiong Deng, Cyrus Shahabi, Ugur Demiryurek, Linhong Zhu, Rose Yu, and Yan Liu. 2016. Latent Space Model for Road Networks to Predict Time-Varying Traffic. In *KDD*. 1525–1534.
[11] Huizhong Duan and ChengXiang Zhai. 2015. Mining coordinated intent representation for entity search and recommendation. In *CIKM*. ACM, 333–342.
[12] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. 2013. Supporting keyword search in product database: a probabilistic approach. *VLDB Endowment* (2013), 1786–1797.
[13] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR* (2011), 2121–2159.
[14] Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. 2017. A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning. In *NIPS*. 3604–3613.
[15] Songwei Ge, Zhicheng Dou, Zhengbao Jiang, Jian-Yun Nie, and Ji-Rong Wen. 2018. Personalizing Search Results Using Hierarchical RNN with Query-aware Attention. In *CIKM*. 347–356.
[16] Samuel Gershman and Noah D. Goodman. 2014. Amortized Inference in Probabilistic Reasoning. In *CogSci*.
[17] Karol Gregor, George Papamakarios, Frederic Besse, Lars Buesing, and Theophane Weber. 2019. Temporal Difference Variational Auto-Encoder. In *ICLR*.
[18] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Yinglong Wang, Jun Ma, and Mohan Kankanhalli. 2019. Attentive Long Short-Term Preference Modeling for Personalized Product Search. (2019), 19.
[19] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Xin-Shun Xu, and Mohan Kankanhalli. 2018. Multi-modal preference modeling for product search. In *ACM Multimedia*. ACM, 1865–1873.
[20] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *WWW*. 507–517.
[21] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative Metric Learning. In *WWW*. 193–201.
[22] Shubhra Kanti Karmaker Santu, Parikshit Sondhi, and ChengXiang Zhai. 2017. On application of learning to rank for e-commerce search. In *SIGIR*. ACM, 475–484.
[23] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *ICLR*.
[24] Rahul Krishnan, Dawen Liang, and Matthew Hoffman. 2018. On the challenges of learning with inference networks on sparse, high-dimensional data. In *AISTATIS*. 143–151.
[25] Guy Lebanon. 2006. Metric Learning for Text Documents. *PAMI* (2006), 497–508.
[26] Shangsong Liang, Zhaochun Ren, Wouter Weerkamp, Edgar Meij, and Maarten De Rijke. 2014. Time-aware rank aggregation for microblog search. In *CIKM*. ACM, 989–998.
[27] Shangsong Liang, Zhaochun Ren, Yukun Zhao, Jun Ma, Emine Yilmaz, and Maarten De Rijke. 2017. Inferring dynamic user interests in streams of short texts for user clustering. *TOIS* 36, 1 (2017), 10.
[28] Shangsong Liang, Xiangliang Zhang, Zhaochun Ren, and Evangelos Kanoulas. 2018. Dynamic Embeddings for User Profiling in Twitter. In *KDD*. 1764–1773.
[29] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* (2008), 2579–2605.
[30] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *SIGIR*. 43–52.
[31] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-Embedding Attributed Networks. In *WSDM*. 393–401.
[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.
[33] Jay Michael Ponte. 1998. *A language modeling approach to information retrieval*. Ph.D. Dissertation.
[34] Qi Qian, Rong Jin, Shenghuo Zhu, and Yuanqing Lin. 2015. Fine-grained visual categorization via multi-stage metric learning. In *CVPR*. 3716–3724.
[35] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.
[36] Alex Rosenfeld and Katrin Erk. 2018. Deep neural models of semantic shift. In *NAACL*. 474–484.
[37] Jennifer Rowley. 2000. Product search in e-shopping: a review and research propositions. *Journal of consumer marketing* 17, 1 (2000), 20–35.
[38] R. Short, II and K. Fukunaga. 2006. The Optimal Distance Measure for Nearest Neighbor Classification. *IEEE Trans. Inf. Theor.* (2006), 622–627.
[39] John Z. Sun, Dhruv Parthasarathy, and Kush R. Varshney. 2014. Collaborative Kalman Filtering for Dynamic Matrix Factorization. *IEEE Trans. Signal Processing* 62, 14 (2014), 3499–3509.
[40] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent Relational Metric Learning via Memory-based Attention for Collaborative Ranking. In *WWW*. 729–739.
[41] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2016. Learning latent vector spaces for product search. In *CIKM*. ACM, 165–174.
[42] Greg Welch and Gary Bishop. 1995. *An Introduction to the Kalman Filter*. Technical Report. Chapel Hill, NC, USA.
[43] Chen Wu, Ming Yan, and Luo Si. 2017. Ensemble methods for personalized e-commerce search challenge at CIKM Cup 2016. *arXiv preprint arXiv:1708.04479* (2017).
[44] Teng Xiao, Shangsong Liang, and Zaiqiao Meng. 2019. Hierarchical Neural Variational Model for Personalized Sequential Recommendation. In *WWW*. ACM, 3377–3383.