Item Cold-Start Recommendations: Learning Local Collective Embeddings

Martin Saveski
Yahoo! Labs, Barcelona
msaveski@yahoo-inc.com

Amin Mantrach
Yahoo! Labs, Barcelona
amantrac@yahoo-inc.com

ABSTRACT

Recommender systems suggest to users items that they might like (*e.g.*, news articles, songs, movies) and, in doing so, they help users deal with information overload and enjoy a personalized experience. One of the main problems of these systems is the item cold-start, *i.e.*, when a new item is introduced in the system and no past information is available, then no effective recommendations can be produced. The item cold-start is a very common problem in practice: modern online platforms have hundreds of new items published every day. To address this problem, we propose to learn *Local Collective Embeddings*–a matrix factorization that exploits items' properties and past user preferences while enforcing the manifold structure exhibited by the collective embeddings. We present a learning algorithm based on multiplicative update rules that are efficient and easy to implement. Experiments on two item cold-start use cases: news recommendation and email recipient recommendation, demonstrate the effectiveness of this approach and show that it significantly outperforms six state-of-the-art methods for item cold-start.

1. INTRODUCTION

Recommender systems are aimed to help users of online platforms to deal with the large volumes of information and to provide them a personalized experience. This is achieved by suggesting items of interest to the users based on their explicit and implicit preferences. Recommender systems use a number of different technologies, but may be broadly classified into two groups: content-based and collaborative filtering systems. Content-based systems examine the properties of the items and recommend items which are similar to the ones the user preferred in the past. They model the taste of a user by building a user profile based on the properties of the items the user liked, and use the profile to compute the similarity with new items. Items which are most similar to the user's profile are recommended. Collaborative filtering systems, on the other hand, ignore the properties of the items and base their recommendations on community preferences. They recommend items that users with similar tastes and preferences liked in the past. Two users are considered similar if they have many items in common.

One of the main problems for recommender systems is the cold-start problem, *i.e.*, when a new item or user is introduced in the system. In this study we focus on the problem of producing effective recommendations for new items: the item cold-start. Collaborative filtering systems suffer from this problem as they rely on the previous ratings of the users. Content based approaches, on the other hand, may still produce recommendations using the description of the items and are the default solution to the item cold-start. However, they tend to achieve lower accuracy and, in practice, they are seldom the only choice.

The problem of item cold-start is of great practical importance because of two main reasons. First, modern online platforms have hundreds of new items everyday and effectively recommending them is essential for keeping the users continuously engaged. Second, collaborative filtering methods are at the core of most recommendation engines, as they tend to achieve the state-of-the-art accuracy [16]. However, to produce recommendations at the expected accuracy they require that items are rated by a sufficient number of users. Therefore, it is crucial for every collaborative recommender to reach this state as soon as possible. Having methods that produce accurate recommendations for new items will allow enough feedback to be collected in a short amount of time, making effective collaborative recommendations possible.

Recently, matrix factorization techniques have been extensively used in the recommendation systems and topic modelling literature. Many collaborative filtering systems approximate the collaborative matrix by applying techniques such as Singular Value Decomposition (SVD) or UV decomposition [20]. Similar matrix factorization techniques have been used to discover topics in a document collections by decomposing the content, *i.e.*, document-term matrix. Non-negative Matrix Factorization (NMF) is one such approach that factorizes the document-term matrix in two non-negative, low-rank matrices, where one matrix corresponds to the topics in the collection and the other represents the extent to which documents belong to these topics. Due to the non-negativity constraints, NMF produces a so-called "additive parts-based" representation of the data that increases the sparsity and interpretability of the hidden factors [12].

In this paper, we propose a new hybrid recommendation approach that exploits both the properties of the items and the similarity of the user preferences. We introduce *Local*

Collective Embeddings (LCE), a collective matrix factorization technique that collectively decomposes the content and the collaborative matrices in a common low-dimensional space while preserving the local geometrical structure of the data.

Given the description of a new item (*e.g.*, the content of a news article), we may project it in the common low-dimensional space and infer the users which are most likely to be interested in it. By doing so, we are able to overcome the item cold-start problem. Moreover, LCE provides a natural way of explaining the recommendations — in terms of user's affinity to topics. Finally, we perform an extensive experimental evaluation of the models on two item cold-start use cases: email recipient recommendation (based on explicit feedback) and news recommendation (based on implicit feedback). We show that the proposed models outperform six state-of-the-art baseline approaches.

Our contributions in this paper can be summarized as follows:

- We introduce a new method for recommendation, LCE, that combines the content and collaborative information in a unified matrix factorization framework while exploiting the local geometrical structure of the data;
- We propose a simple and efficient learning algorithm, based on multiplicative update rules and prove its convergence;
- We conduct an extensive experimental study and we show that the proposed methods outperform six state-of-the-art methods for item-cold start recommendation.

2. RELATED WORK

In this section, we briefly describe several hybrid recommender systems that can handle the item cold-start scenario.

Soboroff [29] proposed a technique based on Latent Semantic Indexing (LSI) for combining the collaborative filtering input and the document content for recommendation of textual items. The method builds a content profile for each user as a linear combination of the preferred documents. LSI is then applied to the user profiles to discover topics in the collection and implicitly learn commonalities among the user profiles. Incoming documents are projected into the LSI space and compared to user profiles. The documents are recommended to the users with the most similar profiles. The author argues that applying LSI on the user profiles instead of the documents allows one to take into account the collaborative input and consequently improves the recommendation performance. However, the system is not evaluated in the cold-start scenario. In section 5, we compare this technique against the method we propose.

Schein *et al.* [26] propose a probabilistic model for cold-start recommendations that is very similar to the one proposed by Soboroff. Their approach extends the work of Hoffman and Puzicha [18] which models the joint distribution of users and items through an aspect model that clusters users and items in a latent space. In order to deal with new items, instead of modelling the joint distribution of users and items, the authors propose to model the joint distribution of users and content features. At query time a "folding-in"' technique [17] is used to embed new items into the latent space so that items can be recommended. After careful analysis one may notice that the technique essentially boils down to building user profiles and applying pLSA to discover latent factors. Taking into account that previous studies have shown the correspondence between pLSA and NMF [15], one may clearly distinguish between this approach and our proposal. Instead of explicitly building user profiles and finding latent features, we discover a latent space common to both the content and collaborative information that allows us to link one to the other.

Singh and Gordon [28] propose the idea of collective matrix factorization, a general framework for multi-relational factorization models. They subsume models on any number of relations as long as their loss function is a twice differentiable decomposable loss. In their work, they address both rating prediction and item recommendation. The matrix factorization approach proposed in this work is based on a similar idea of collective factorization. However, we enforce non-negativity constraints on the factorization to obtain sparse and interpretable factors, and we consider the specific scenario of cold-start recommendations.

Shmueli *et al.* [27] consider a similar scenario of news recommendation (Section 5.3), *i.e.*, predicting the articles a user is most likely to comment on. They combine content-based and collaborative filtering approach using a latent factor model. The odds that a user will comment an article are estimated as the inner product of the user and article factors, where the article factors are represented as the sum of the latent factors associated with the textual content (tags–named entities) and the commenters. A modification of the model for real-time scenarios is presented in [3]. The authors show that the recommendation accuracy grows as the number of commenters grows. However, in an item cold-start scenario articles are not yet associated with commenters, thus an article can only be represented with the latent factors of the textual tags.

Exploiting the local geometric structure of the data to discover better low-dimensional representations has been exploited by Cai *et al.* [9]. Inspired by the success of using the nearest neighbour graph for label propagation in semi-supervised learning, they propose a clustering technique. The algorithm favours factorizations for which similar instances have similar low-dimensional representations. The authors show that, by imposing this constraint, they outperform classical clustering algorithms and classical factorization techniques. In this work, we impose such geometrical constraints but for collective factorization for which we can handle multiple data sources, *i.e.*, the content and collaborative data matrices.

3. LOCAL COLLECTIVE EMBEDDINGS

In this section we formally define the problem, we explain the intuition behind learning collective embeddings and exploiting locally, and finally we show how such embeddings can be learnt and how they can be used for prediction.

3.1 Problem Statement

The scenario we consider is the item cold-start recommendation, where we would like to suggest new items – for which no interests has been expressed so far – to potentially interested users. Given a new item, its corresponding description and the patterns of past activities of the users, we want to retrieve users who would likely manifest interest in this item. More formally, we can define the problem as follows. At training time, we are given a collection of n items described by: (1) a set of m properties stored in a matrix $\mathbf{X_s} \in \mathbb{R}^{n \times m}$, where a row corresponds to an item and a column to an item property; and (2) a set of u users stored in a matrix $\mathbf{X_u} \in \mathbb{R}^{n \times u}$, where a cell (i, j) indicates if the user j has shown interest in item i. At test time, we are given a new item $\mathbf{q}$ with description $\mathbf{q}_s \in \mathbb{R}^{1 \times m}$, and our goal is to predict $\mathbf{q}_u \in \mathbb{R}^{1 \times u}$, *i.e.*, to score how likely is a user to show interest in the new item.

3.2 Intuition Behind Collective Factorization

Given the problem defined, items are associated with a description and a set of users who consumed them. In the case of news, each news article is described by the set of words it contains and by all the users that commented on it. This information is then represented with two matrices, a document-term matrix $\mathbf{X_s} \in \mathbb{R}^{n \times v}$, and a document-user matrix $\mathbf{X_u} \in \mathbb{R}^{n \times u}$, where n is the number of documents, v is the vocabulary size and u is the number of users. The document-term matrix ($\mathbf{X_s}$) may be a boolean matrix or may represent the TF-IDF scores of the words in the document. On the other hand, the entries of the document-user matrix ($\mathbf{X_u}$) reflect whether a given user commented on a given article. If we factorize $\mathbf{X_s}$ in two lower-dimensional matrices, we will discover the topics that appear in the documents and the extent to which each document belongs to these topics. Similarly, factorizing $\mathbf{X_u}$ leads to the discovery of user communities and the extend to which each document triggers interest within the communities. However, if factorized independently each factorization will represent a different latent space and there will be no correspondence between the topics and the communities. The idea of LCE is that both, documents and users, should be represented in a common latent space. In other words, each factor can be described by a set of words (*i.e.*, a topic) but also by a set of users (*i.e.*, a community). To achieve this, we collectively factorize $\mathbf{X_s}$ and $\mathbf{X_u}$ and enforce a low-dimensional representation in a common space. Additionally, to achieve an additive effect we impose non-negativity constraints that lead to interpretable and sparse latent representations.

More formally, given the matrices $\mathbf{X_s}$ and $\mathbf{X_u}$, we define the following optimization problem:

$$\min : J = \frac{1}{2}[\alpha||\mathbf{X_s} - \mathbf{W}\mathbf{H_s}||^2 + (1-\alpha)||\mathbf{X_u} - \mathbf{W}\mathbf{H_u}||^2 + \\ + \lambda(||\mathbf{W}||^2 + ||\mathbf{H_s}||^2 + ||\mathbf{H_u}||^2)] \quad (1)$$

$$s.t. \quad \mathbf{W} \geq \mathbf{0}, \mathbf{H_s} \geq \mathbf{0}, \mathbf{H_u} \geq \mathbf{0}$$

The first two terms correspond to the factorization of the matrices $\mathbf{X_s}$ and $\mathbf{X_u}$. The common latent space representation is achieved by using the same matrix $\mathbf{W}$ in the decompositions of both $\mathbf{X_s}$ and $\mathbf{X_u}$. $\alpha \in [0, 1]$ is a hyper-parameter that controls the importance of each factorization. Setting $\alpha = 0.5$ gives equal importance to both factorizations, while values of $\alpha > 0.5$ (or $\alpha < 0.5$) give more importance to the factorization of $\mathbf{X_s}$ (or $\mathbf{X_u}$). The remaining terms are Tikhonov (Frobenius norm) regularization of $\mathbf{W}$, $\mathbf{H_u}$, and $\mathbf{H_s}$, controlled by the hyper-parameter $\lambda \geq 0$. It is used to enforce smoothness of the solution and avoid overfitting.

3.3 Exploiting Locality

When performing collective factorization, as in Eq. (1), we attempt to find a common low-dimensional space that is optimized for the linear approximation of the data from both views. We make an implicit assumption that the data from both views is drawn from a common distribution. One may hope that additional knowledge of this distribution can be exploited to discover a better low-dimensional space. A natural assumption could be that: if two data points x_i and x_j, in any view, are close in the intrinsic geometry of the distribution, then their representations in the low-dimensional space should also be close to each other. This assumption is commonly referred to as the *manifold assumption* and plays an essential role in algorithms for dimensionality reduction [6] and semi-supervised learning [7, 30].

In reality the geometric structure of the distribution is not known and cannot be directly used. However, recent studies on spectral graph theory [10] and manifold learning [5] have shown that the local geometric structure can be effectively modeled through a nearest neighbor graph on a scatter of data points. Consider a graph with n nodes where each node represents a data point. For each point we find the p nearest neighbors and we connect the corresponding nodes in the graph. The edges may be binary (1 if one of the nearest neighbors, 0 otherwise) or may be weighted (e.g., cosine similarity). This results in a matrix $\mathbf{A}$ which can later be used to measure the local closeness of two points x_i and x_j.

Recall that the collective factorization maps each data point x_i into a low-dimensional representation w_i (a row of the matrix $\mathbf{W}$). A natural way to measure the distance between two low dimensional representations, given the choice of a loss function, is to compute the Euclidean distance: $||w_i - w_j||^2$. Using the above defined weight matrix $\mathbf{A}$ we may measure the smoothness of the low dimensional representation as:

$$\begin{aligned} S &= \frac{1}{2}\sum_{i,j=1}^{n} ||w_i - w_j||^2 \mathbf{A}_{ij} \\ &= \sum_{i=1}^{n}(w_i^T w_i)\mathbf{D}_{ii} - \sum_{i,j=1}^{n}(w_i^T w_j)\mathbf{A}_{ij} \\ &= \mathrm{Tr}(\mathbf{W}^{\mathrm{T}}\mathbf{D}\mathbf{W}) - \mathrm{Tr}(\mathbf{W}^{\mathrm{T}}\mathbf{A}\mathbf{W}) = \mathrm{Tr}(\mathbf{W}^{\mathrm{T}}\mathbf{L}\mathbf{W}), \end{aligned}$$

where $\mathbf{D}$ is a diagonal matrix whose entries are the row sums of $\mathbf{A}$ (or column, as $\mathbf{A}$ is symmetric), *i.e.*, $\mathbf{D}_{ii} = \sum_i \mathbf{A}_{ij}$; $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is called the Laplacian matrix of the graph [10] and $\mathrm{Tr}(\bullet)$ is the trace operator.

3.4 LCE Optimization Problem

Given the above, we modify the formulation of in Eq. (1) as to enforce locality when discovering the factors. This leads to the following optimization problem:

$$\min : J = \frac{1}{2}[\alpha||\mathbf{X_s} - \mathbf{W}\mathbf{H_s}||^2 + (1-\alpha)||\mathbf{X_u} - \mathbf{W}\mathbf{H_u}||^2 + \\ + \beta\mathrm{Tr}(\mathbf{W}^{\mathrm{T}}\mathbf{L}\mathbf{W}) + \lambda(||\mathbf{W}||^2 + ||\mathbf{H_s}||^2 + ||\mathbf{H_u}||^2)]$$

$$s.t. \quad \mathbf{W} \geq \mathbf{0}, \mathbf{H_s} \geq \mathbf{0}, \mathbf{H_u} \geq \mathbf{0}, \quad (2)$$

where $\mathbf{L}$ is the Laplacian matrix of the graph, and β is a hyper-parameter which controls the extent to which locality is enforced. The hyper-parameters α and λ have the same semantics as in Eq. (1).

3.5 Learning Algorithm

The optimization problem defined above is non-convex in terms of all parameters ($\mathbf{W}$, $\mathbf{H_s}$, $\mathbf{H_u}$) together. Thus, it is unrealistic to expect an algorithm to find the global minimum. In what follows, we derive an iterative algorithm based on multiplicative update rules which can achieve a stationary point.

The partial derivatives of J w.r.t. $\mathbf{W}$, $\mathbf{H_s}$, and $\mathbf{H_u}$ are:

$$\nabla_{\mathbf{W}} J = \alpha \mathbf{W}\mathbf{H_s}\mathbf{H_s}^{\mathrm{T}} - \alpha \mathbf{X_s}\mathbf{H_s}^{\mathrm{T}} + (1-\alpha)\mathbf{W}\mathbf{H_u}\mathbf{H_u}^{\mathrm{T}} - (1-\alpha)\mathbf{X_u}\mathbf{H_u}^{\mathrm{T}} + \beta \mathbf{L}\mathbf{W} + \lambda \mathbf{W}, \tag{3}$$

$$\nabla_{\mathbf{H_s}} J = \alpha \mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{H_s} - \alpha \mathbf{W}^{\mathrm{T}}\mathbf{X_s} + \lambda \mathbf{H_s} \tag{4}$$

$$\nabla_{\mathbf{H_u}} J = (1-\alpha)\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{H_u} - (1-\alpha)\mathbf{W}^{\mathrm{T}}\mathbf{X_u} + \lambda \mathbf{H_u} \tag{5}$$

Applying the Karush-Kuhn-Tucker (KKT) first-order optimality conditions to J [11], we derive:

$$\mathbf{W} \geq \mathbf{0}, \quad \mathbf{Hs} \geq \mathbf{0}, \quad \mathbf{Hu} \geq \mathbf{0}, \tag{6}$$

$$\nabla_{\mathbf{W}} J \geq \mathbf{0}, \quad \nabla_{\mathbf{H_s}} J \geq \mathbf{0}, \quad \nabla_{\mathbf{H_u}} J \geq \mathbf{0}, \tag{7}$$

$$\mathbf{W} \odot \nabla_{\mathbf{W}} J = \mathbf{0}, \quad \mathbf{Hs} \odot \nabla_{\mathbf{H_s}} J = \mathbf{0}, \quad \mathbf{Hu} \odot \nabla_{\mathbf{H_u}} J = \mathbf{0}, \tag{8}$$

where $\odot$ corresponds to the element-wise matrix multiplication operator.

Substituting the derivatives of J from Equations (3), (4) and (5) in Equation (8) leads to the following update rules:

$$\mathbf{W} \leftarrow \mathbf{W} \odot \frac{[\alpha \mathbf{X_s}\mathbf{H_s}^{\mathrm{T}} + (1-\alpha)\mathbf{X_u}\mathbf{H_u}^{\mathrm{T}} + \beta \mathbf{A}\mathbf{W}]}{[\alpha \mathbf{W}\mathbf{H_s}\mathbf{H_s}^{\mathrm{T}} + (1-\alpha)\mathbf{W}\mathbf{H_u}\mathbf{H_u}^{\mathrm{T}} + \beta \mathbf{D}\mathbf{W} + \lambda \mathbf{W}]}, \tag{9}$$

$$\mathbf{H_s} \leftarrow \mathbf{H_s} \odot \frac{[\alpha \mathbf{W}^{\mathrm{T}}\mathbf{X_s}]}{[\alpha \mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{H_s} + \lambda \mathbf{H_s}]}, \tag{10}$$

$$\mathbf{H_u} \leftarrow \mathbf{H_u} \odot \frac{[(1-\alpha)\mathbf{W}^{\mathrm{T}}\mathbf{X_u}]}{[(1-\alpha)\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{H_u} + \lambda \mathbf{H_u}]}, \tag{11}$$

where $\frac{\bullet}{\bullet}$ denotes the element-wise matrix division operator. We define the following theorem:

THEOREM 1. *The objective function J in Equation (2) is nonincreasing under the update rules in Equations (9), (10), and (11). The objective function J is invariant under these updates if and only if $\mathbf{H_u}$, $\mathbf{H_s}$ and $\mathbf{W}$ are at a stationary point of the function.*

A detailed proof of the above theorem is provided as supplement material[1].

3.6 Inference

Once the model has been trained to learn $\mathbf{W}$, $\mathbf{H_s}$ and $\mathbf{H_u}$, we can use these factors for prediction. For instance, given the bag-of-words vector of a new news article $\mathbf{q}_s$, we can predict the users that are most likely to leave a comment, *i.e.*, $\mathbf{q}_u$. To do so, we project the document vector $\mathbf{q}_s$ to the common hidden space by solving the overdetermined system $\mathbf{q}_s = \mathbf{w}\mathbf{H_s}$ using the least squares method (with a projection to 0 of the negative values, see [8]). The vector $\mathbf{w}$, computed online, captures the factors — in the common hidden space — that explain the observed news article $\mathbf{q}_s$. Then, by using this low dimensional vector $\mathbf{w}$ we may infer the missing part of the query: $\mathbf{q}_u \leftarrow \mathbf{w}\mathbf{H_u}$. Each element of $\mathbf{q}_u$ represents a score of how likely it is that the user will comment the new article. Then, given these scores, we may rank the users.

[1] https://github.com/msaveski/LCE/blob/master/Th1Proof.pdf

4. EXPLAINING RECOMMENDATIONS

Good recommendations are not only accurate but also transparent, *i.e.*, supported with explanations. This allows the end users to understand the reasoning behind the recommendations and helps them build trust towards the system.

LCE provides a natural way of explaining recommendations in terms of users' affinity to topics. To obtain the topical interest profile for a user i we can construct a vector $\mathbf{x_u} \in \mathbb{R}^{1\times u}$ (u is the number of users), where all elements are equal to zero except $[\mathbf{x_u}]_i = 1$, and solve for $\mathbf{w}$ in $\mathbf{x_u} = \mathbf{w}\mathbf{H_u}$; every element of $\mathbf{w}$ quantifies the user's affinity to a specific topic. Topics may be presented using the top-k terms or by automatic annotation (*e.g.*, [21]). Furthermore, by computing $\mathbf{x_s} \leftarrow \mathbf{w}\mathbf{H_s}$ ($\mathbf{x_s} \in \mathbb{R}^{1\times m}$), we obtain the association between the user and every word in the vocabulary; we may present this information to the user, *e.g.*, using a word cloud.

To debug and track down the source of unexpected behaviour of the system one may examine the link between topics and communities (*e.g.*, by looking at the top words and users). This link may also be exploited in other applications, such as advertising, where advertisers can easily identify target users based on their topical interests.

5. EXPERIMENTAL EVALUATION

In this section we present a series of experiments to evaluate the performance of LCE in the item cold-start scenario. We first describe the baselines and then we compare them with LCE in two item cold-start use cases: email recipient recommendation and news recommendation. Finally, we analyse the parameter settings and the running times.

5.1 Baselines for Comparison

We compare LCE to six other approaches: pure content-based recommender, content-topic-based recommender, LSI applied on the author profiles, author topic-model, learning attribute-to-feature mappings, and fLDA.

Content-based Recommender (CB). We build a profile for each user based on the properties of the items preferred in the past. Experimentally we find that weighting each item inversely proportional to the number of users that interacted with the item leads to an improved performance. Thus, in the user profile, very popular items are given less importance, while less popular items are given more importance. More formally, a user profile U is defined as: $U = \sum_{i\in I}(\vec{v_i}/freq_i)$, where I is the set of items the user interacted with in the past, $\vec{v_i}$ is the description of item i and $freq_i$ is the number of users that interacted with i. At test time, we rank the items by computing the cosine similarity between the new items and the user profiles.

Content-topic-based Recommender (CTB). We extract topics from the content of the items by applying NMF and we describe each item as a mixture of the topics extracted. We then build a topical profile for each user based on the topics of the items the user interacted with in the past. At test time, we infer the topics of the new items and we rank the items based on the cosine similarity between the item's topics and the users' topical profiles. The CTB recommender allows us to investigate the importance of performing joint factorization of both the content and collaborative matrix, instead of factorizing only the content matrix.

LSI on the User Profiles (UP-LSI). We apply the hybrid recommendation system proposed in [29] (see Section 2). The approach combines the content and collaborative information by building user profiles and applying Latent Semantic Indexing (LSI) to discover latent factors. At test time, the new items are projected in the latent space and compared to the user profiles. Finally, the items are recommended to the users with the most similar profiles.

Author-topic Model (ATM). The author-topic model [24] is a generative probabilistic model which extends LDA to include authorship information. It associates each author with a multinomial distribution over topics, and each topic with multinomial distribution over words. As the authors point out, the model may not only be used to find the topics associated with the authors, but also to predict the authors of unobserved documents. In the email recipient recommendation experiment we model the recipients as authors, while in the news recommendation scenario we model the users as authors. As recommended, we set the parameters as: $\alpha = 50/k$, where k is the number of topics, $\beta = 0.01$, and we perform 500 iterations of the Gibbs Sampler.

Learning Attribute-to-Feature Mappings (BPR-kNN). This method [14] handles the cold-start in two steps: (1) factorizing the collaborative matrix to learn latent factor representation of the users and items, (2) learning a mapping between the user/item attributes and the corresponding latent factors. When a new user/item arrives in the system, the mapping from (2) is used to infer the factors from the attributes, and then the factors are used to make predictions. As proposed by the authors, we used Bayesian Personalized Ranking (BPR) to factorize the collaborative matrix. To learn the mappings we used the K-Nearest-Neighbour and BPR optimization; however, the kNN mapping was superior in all cases and thus we report the results for only for kNN mapping. This is consistent with their experimental results. We test with different number of latent factors $k \in \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$.

fLDA. Is a latent factor model proposed in [2] and it is an extension of the Regression Based Latent Factor model [1] for applications where "bag-of-words" representation of items is natural. The main idea is regularizing the user and item factors simultaneously through the user features and the words associated with the items. The user ratings are modelled as user's affinity to the item's topics, where the user's affinity to topics and topic assignments to items are learned jointly in supervised fashion. In our experiments we only use the item feature as user features are not available. As recommended by the authors, we run 20 EM iterations with 100 samples, drawn after 10 burn-in samples. In the email recommendation experiment, we test for $k \in \{10, 25, 50\}$ factors, while for the news recommendation experiment we test only $k = 10$ due to the long running times.

5.2 Email Recipient Recommendation

When people write emails they do not necessarily start by filling in the recipient address (*i.e.*, the "to" field), but may start by writing the body of the message. Given the content of the message and the messaging habits of the user, *i.e.*, with whom the user exchanged messages with similar content in past, we would like to predict the most likely recipients of the new message. In this experiment, we test the accuracy of the recipient recommendations produced by LCE against the six baseline techniques.

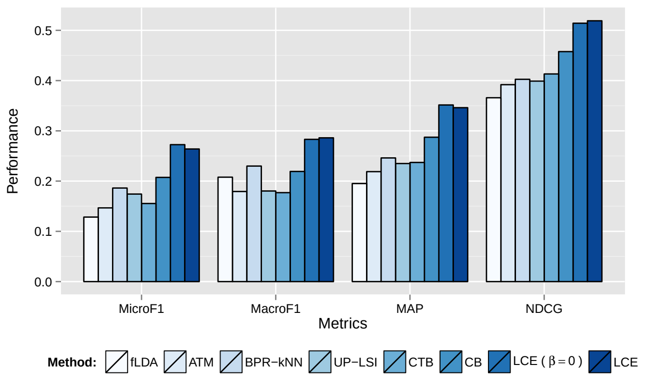

Figure 1: Comparison of the different methods on the Enron dataset.

Dataset. The data is composed of email messages released during investigation of the Federal Energy Regulatory Commission against the Enron Corporation. We consider the 10 largest mailboxes and within each mailbox the emails sent by the owner. There are 36,010 emails sent to 4,984 recipients. The size of the vocabularies for each mailbox ranges from 12,375 to 56,193 unique tokens. The messages have been preprocessed by removing the headers (from/to/cc fields), converting all tokens to lower case and removing numbers, stop-words and infrequent tokens (appearing < 5 times).

Evaluation Metrics. The output of the algorithms is a ranking of the past recipients of how likely they are to be recipients of the new email. The feedback from the users is explicit, *i.e.*, we have ground truth of who are the recipients of the mail as specified by the user. To evaluate the ranking produced by each algorithm we use the state-of-the-art metrics from Information Retrieval: Micro and Macro F1, Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (NDCG) [4].

Evaluation Protocol. As the data is intrinsically influenced by time we sort the mailboxes chronologically. We divide the messages in 80% training and 20% testing, resulting in 10 independent train/test subsets. Only the recipients that appear in the training period are considered as potential receivers. We tune the hyper-parameters of the methods on independent validation set, 10% of the training set. Finally, we evaluate the statistical significance of the differences in performance by using a Wilcoxon signed rank test [13].

Results. Figure 1 shows the average performance of each method across the 10 mailboxes. LCE performs better than the other methods in all measures with differences ranging from 5%-15%. All differences are statistically significant (Wilcoxon signed rank test, $p < 0.05$). Imposing locality leads to small performance improvements in some measures, however the differences are not statistically significant. This indicates that nearest neighbour graph does not bring additional information in the case of emails. One explanation may be that emails are user generated content and as such contain a lot of noise, such as misspellings or informal expressions, that leads to inaccurate nearest neighbour graphs.

5.3 News Recommendation

To improve the user experience, online news platforms allow users to engage with articles by posting comments. Moreover, to encourage user engagement on the platform the users are recommended articles that they may be interested in. We consider the item cold-start scenario, *i.e.*, when a new article is published and none of the users have commented on it yet. Thus, given the content of the new articles and the past commenting patterns we would like to recommend to the users the articles that they are most likely to comment.

Dataset. We consider a random sample of news articles and the corresponding comments posted on the Yahoo! News website in a period of 40 days. The dataset contains ~41K articles, ~3.5M comments posted by ~650K users. The size of the vocabulary is ~60K (*i.e.*, number of unique tokens in all articles) and ~9M tokens. The content of the articles is preprocessed such that all tokens are converted to lower case, and stop-words, digits, punctuation, short (< 3 characters) and infrequent (appearing < 3 times) tokens are removed.

Evaluation Metrics. Similar to the previous experiment, the output of each algorithm is a ranking. In this experiment, however, we do not have an explicit feedback of which news articles were undesired by the users. While, commenting an article is an evidence of the user's interest in it, the absence of a comment is not an indication that the article was undesired, as not commenting may stem from multiple different reasons. Therefore, we adopt the average percentile ranking, a measure proposed in [19] and widely used to evaluate ranking based on implicit feedback (e.g., [23, 25]). We define $rank_{u,i}$ as the percentile ranking of article i in the ranked list of articles for the user u; if $rank_{u,i} = 0\%$, then the article i is predicted to be the most interesting for u, while $rank_{u,i} = 100\%$ implies that the article is predicted to be the least interesting. Our quality measure is then the total average percentile ranking of an article:

$$\overline{rank} = \frac{\sum_{u,i} comment_{u,i} \cdot rank_{u,i}}{\sum_{u,i} comment_{u,i}},$$

where $comment_{u,i}$ is an indicator function that equals to: 1 if the user u commented on article i; and 0 otherwise. The lower $\overline{rank}$, the better the quality of the ranking. For random predictions, the expected value of $\overline{rank}$ is 50%. Thus, if $\overline{rank} < 50\%$, then the algorithm is better than random. To ease illustration, we convert the percentile ranking into *ranking accuracy* (RA). That is 1 (best/ideal predictions), if the percentile ranking is 0%; and it is 0 (random predictions), if the percentile ranking is 50%:

$$Ranking\ Accuracy = \frac{50\% - \overline{rank}}{50\%}.$$

We evaluate the Ranking Accuracy (RA) at different positions: 3, 5, 7 and 10.

Evaluation Protocol. We sort the data chronologically and we produce train/test subsets by shifting a time window. We train using the past 30 days and we predict comments on the next day, shifting for one day at a time, resulting in 10 independent folds. We also restrict our test set to those users who have commented at least once in the training period. We tune the hyper-parameters of each method on an independent validation set, 10% of the training set, and we evaluate the statistical significance of the differences in performance by using Wilcoxon signed rank test [13].

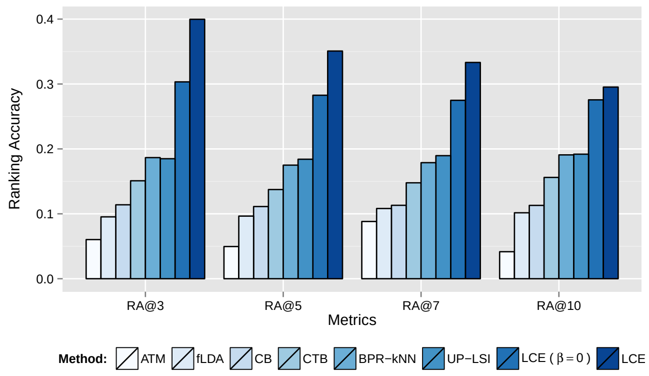

Figure 2: Comparison of the different methods on the Yahoo! News dataset.

Results. We evaluate the different methods in each of the 10 testing days and we compute the average performance (Figure 2). All algorithms perform better then random, *i.e.*, RA $> 0\%$. LCE outperforms all other methods with statistically significant differences (Wilcoxon signed rank test, $p < 0.05$). Imposing locality leads to better ranking accuracy in all positions, however, the effect diminishes as we consider larger lists. The difference is statistically significant only for RA@3 and RA@5 (Wilcoxon signed rank test, $p < 0.05$). This indicates that exploiting the manifold structure of the data allows the algorithm to push the relevant items towards the top of the list. As users are usually presented a short list of recommendations, making accurate recommendations on the top of the list is crucial for improving the satisfaction of the users.

5.4 Parameter Analysis

The LCE model has three essential parameters: k, number of latent variables, *i.e.*, topics/communities; α, weight of the content versus the collaborative information; and λ, controlling the smoothness of the solution. Figure 3 shows a typical behaviour of the algorithm for different values of the parameters. The results are averaged over 10 runs of all algorithms on one mailbox from the Enron dataset.

The parameter k controls the complexity of the model. Small values of k, *i.e.*, simple models under-fit whereas large values of k over-fit the data and lead to poor performance (Figure 3, left). Thus, one has to find a balance between the two that fits best the problem at hand. Furthermore, balancing the importance of the content versus the collaborative information, *i.e.*, $\alpha \approx 0.5$ tends to achieve the best performance. Figure 3 (middle) suggests that giving slightly more importance to the collaborative information (*e.g.*, $\alpha \in [0.2, 0.5]$) may be helpful. Finally, adding the Tikhonov regularization helps; however, large values of λ, oversimplify the model and decrease performance. Setting $\lambda \in (0, 1)$ leads to stable and high performance (Figure 3, right).

The nearest neighbor graph ($\mathbf{A}$) in LCE may be constructed using the content or the collaborative information. The weights associated with the neighbors may be binary or the cosine similarity between the documents. We find that imposing the graph regularization based on the collaborative information leads to better performance (Figure 4). The bi-

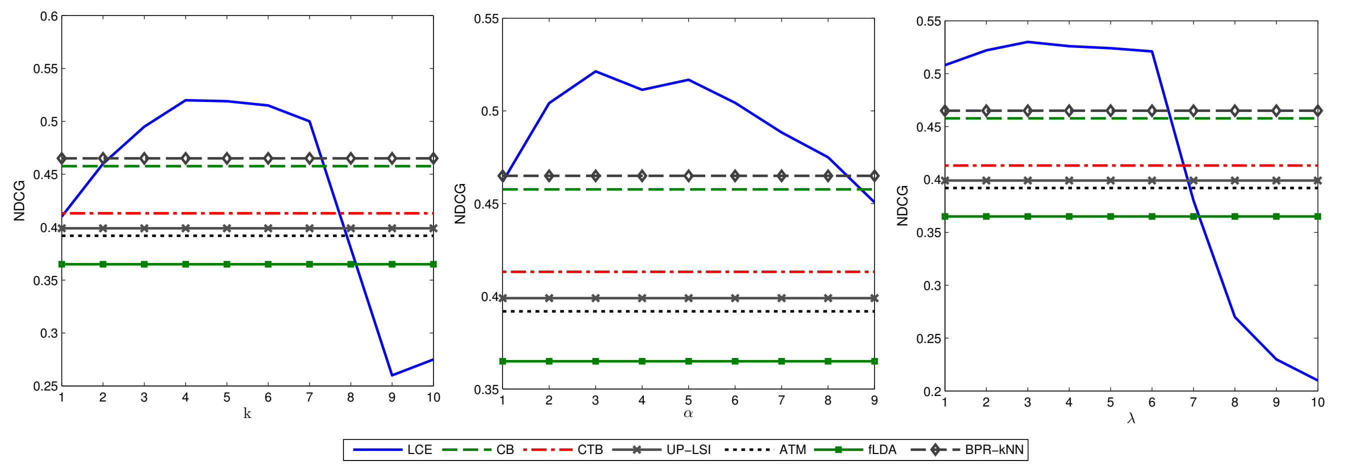

Figure 3: Behaviour of the LCE hyper-parameters ($\beta = 0.25$). Left: k, number of topics; Middle: α, weight of the content versus the collaborative information; and λ, the smoothness of the solution. The results are averaged over 10 runs of the methods on Tana Jones' mailbox from the Enron data set.

nary weighting scheme is more sensitive to the number of nearest neighbors (p) and works better for small p, while the cosine weighting scheme, as expected, is less sensitive to the choice of p.

The parameter β behaves similarly to λ and setting $\beta = 0.25$ leads to higher performance (for brevity we do not report results for different values of β).

5.5 Running Time Analysis

In this section, we measure the CPU time required by the methods under different settings of the model complexity. All models are comparably fast at inference time, as they require only simple operations such as projections in the latent space or computing similarities. Hence, we only report the running times needed to train the models.

For this experiment, we consider one mailbox from the Enron dataset (4K messages, 500 recipients, 18K unique terms) on which we perform 10 runs of each method under different values of the hyper-parameter k. The averaged CPU times are reported in Figure 5. Due to the long computation time required, we test the author-topic model up to $k = 500$ and fLDA up for $k \in \{10, 25, 50\}$. In the case of UP-LSI k is bounded by 500 due to the rank of the matrix.

The content-based recommender (CB) takes least time for training as it only requires building the user profiles. Little time is also required by the UP-LSI method relying on a fast sparse SVD implementation. ATM and fLDA are computationally most expensive; ATM requires between 3 and 35 hours to train, and fLDA requires between 5 and 46 hours to train even for small values of k. The LCE (with $\beta = 0$, *i.e.*, without building kNN graph) and BPR-kNN have similar running times. The LCE with $\beta = 0.25$, on the other hand, is slightly slower than other methods, except for ATM and fLDA. LCE is reasonably fast and requires 25 minutes to train for the highest values of k, suggesting that frequent updates of the model are possible.

5.6 Reproducibility of the Experiments

The Matlab implementations of the LCE are made publicly available at: `https://github.com/msaveski/LCE`. As

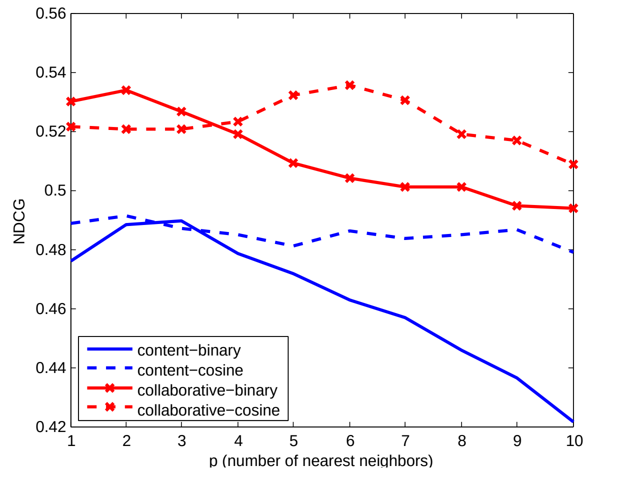

Figure 4: Different ways of constructing the nearest neighbor graph in LCE. The results are averaged over 10 runs on Tana Jones' mailbox.

discussed in Section 5.4, the parameters may be set as: $\alpha = 0.5$, $\beta = 0.25$, and $\lambda = 0.5$, while the parameter k depends on the data and needs to be tuned. A Matlab implementation of the Author-topic Model is publicly available as part of the Matlab Topic Modeling Toolbox at: `http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm`. An R and C/C++ implementation of fLDA is available at: `https://github.com/beechung/Latent-Factor-Models`. An implementation of BPR-kNN is available at: `https://github.com/zenogantner/MyMediaLite`. Finally, the Enron dataset is available at: `https://www.cs.cmu.edu/~enron/`. In the experiments, we consider the 10 largest mailboxes owned by: Steven Kean, Vince Kaminski, Jeff Dasovich, Sally Beck, Tana Jones, Mark Haedicke, Sara Shackleton, Mark Taylor, John Lavorato, and Louise Kitchen. The parameter and run time analysis experiments are performed on the mailbox of Tana Jones.

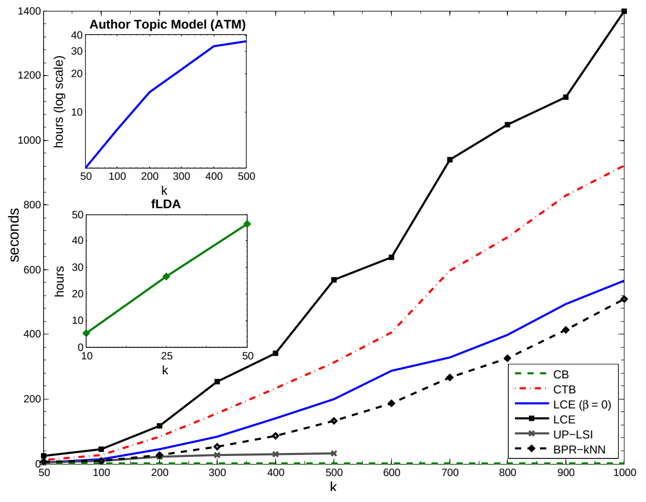

Figure 5: Running time required to train each method. We report the CPU times in seconds averaged over 10 runs on Tana Jones' mailbox.

6. CONCLUSIONS AND FUTURE WORK

To overcome the item cold-start, in this work we have proposed LCE, a recommender system that combines content and collaborative information in a unified matrix factorization framework. Our proposed algorithm outperform existing item-cold start recommenders. Interestingly, in case of rich content (*e.g.*, news articles) imposing locality to exploit the manifold structure of the data improves the ranking accuracy in the top positions of the rankings which is crucial for improving the user satisfaction on news platforms.

Towards Online Adaptive Models. In some scenarios it is desired that the models are updated as new data is arriving. In such context, time plays in important role when modeling user preferences. In this line of research, recently, McAuley *et al.* [22] modeled user tastes evolution and showed an improvement in recommendations. To our knowledge none of the existing "time-aware" approaches have been applied to the item cold-start recommendations. As a future work we consider extending our models to close this gap.

7. REFERENCES

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *International conference on Knowledge Discovery and Data mining*, 2009.

[2] D. Agarwal and B.-C. Chen. flda: Matrix factorization through latent dirichlet allocation. In *International Conference on Web Search and Data Mining*, 2010.

[3] M. Aharon, A. Kagian, R. Lempel, and Y. Koren. Dynamic personalized recommendation of comment-eliciting stories. In *ACM conference on Recommender systems*, 2012.

[4] R. Baeza-Yates and Ribeiro-Neto. *Modern information retrieval*. ACM press New York, 1999.

[5] M. Belkin. *Problems of learning on manifolds*. PhD thesis, The University of Chicago, 2003.

[6] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems*, 2001.

[7] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 2006.

[8] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 2007.

[9] D. Cai, X. He, X. Wu, and J. Han. Non-negative matrix factorization on manifold. In *International Conference on Data Mining*, 2008.

[10] F. Chung. *Spectral Graph Teory*. AMS, 1997.

[11] A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. Wiley, 2009.

[12] L. Daniel and S. Sebastian. Learning the parts of objects by non-negative matrix factorization. *Nature*, 1999.

[13] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 2006.

[14] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *International Conference on Data Mining*, 2010.

[15] E. Gaussier and C. Goutte. Relation between plsa and nmf and implications. In *Special Interest Group on Information Retrieval*, 2005.

[16] A. T. Gediminas Adomavicius. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 2005.

[17] T. Hofmann. Probabilistic latent semantic indexing. In *Special Interest Group on Information Retrieval*, 1999.

[18] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *International Joint Conferences on Artificial Intelligence*, 1999.

[19] Y. Hu, Y. Koren, and C. Volinsky. Collaborative Filtering for Implicit Feedback Datasets. In *International Conference on Data Mining*, 2008.

[20] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.

[21] J. H. Lau, K. Grieser, D. Newman, and T. Baldwin. Automatic labelling of topic models. In *Annual Meeting of the Association for Computational Linguistics*, 2011.

[22] J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *World Wide Web Conference*, 2013.

[23] D. Quercia, N. Lathia, F. Calabrese, G. Di Lorenzo, and J. Crowcroft. Recommending social events from mobile phone location data. In *International Conference on Data Mining*, 2010.

[24] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *Uncertainty in Artificial Intelligence*, 2004.

[25] D. Saez-Trumper, D. Quercia, and J. Crowcroft. Ads and the city: considering geographic distance goes a long way. In *ACM conference on Recommender systems*, 2012.

[26] A. Schein, A. Popescul, L. Ungar, and D. Pennock. Methods and metrics for cold-start recommendations. In *Special Interest Group on Information Retrieval*, 2002.

[27] E. Shmueli, A. Kagian, Y. Koren, and R. Lempel. Care to comment?: recommendations for commenting on news stories. In *World Wide Web Conference*, 2012.

[28] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *ACM Conference on Knowledge Discovery and Data Mining*, 2008.

[29] I. Soboroff. Combining content and collaboration in text filtering. In *IJCAI Workshop on Machine Learning for Information Filtering*, 1999.

[30] X. Zhu and J. Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *International Conference on Machine learning*, 2005.