Sage: Recommender Engine as a Cloud Service

Royi Ronen, Noam Koenigstein, Elad Ziklik, Mikael Sitruk, Ronen Yaari, Neta Haiby-Weiss
Microsoft Israel
{royir,noamko,eladz,mikaels,royaari,netahw}@microsoft.com

ABSTRACT

Project Sage is Microsoft's all-purpose recommender system, designed and deployed as an ultra-high scale cloud service. Sage focuses on both state of the art research and high scale robust implementation. In the research front, we demonstrate new pre-processing and cleaning techniques, a novel probabilistic matrix factorization model for implicit one-class data, and a relatively new evaluation framework. In the engineering front, we present a working service deployed on the Microsoft Azure cloud, which provides easy-to-use interfaces to integrate a recommendation service into any website.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Data Applications - *Data Mining*.

Keywords

Recommender Systems, Cloud Services

1. INTRODUCTION

Personalized recommendation systems are becoming increasingly popular in many websites and online stores. While there is a clear demand for such systems, there are only few high-scale implementations to choose from. A notable example is *Apache Mahout*, which provides source code for several well-established recommendation algorithms. As opposed to Mahout, Sage is a live service on the Microsoft Azure cloud, which exposes easy-to-use interfaces and a dashboard to design and train models, as well as to explore personalized recommendations and item-to-item relations. Our service integrates several novel algorithms, to form an all-purpose recommender, based on one-class implicit feedback.

Figure 1 depicts our system overview. There are 5 main stages: (1) Data collection; (2) Data cleaning and feature selection; (3) Variational Bayes matrix factorization; (4) Qualitative and quantitative evaluation; (5) Online service.

RecSys'13, October 12–16, 2013, Hong Kong, China.
ACM 978-1-4503-2409-0/13/10.
http://dx.doi.org/10.1145/2507157.2508221.

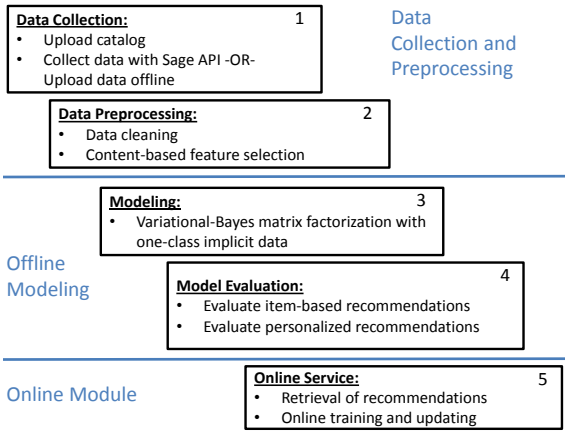

Figure 1: Sage System Overview

2. SAGE OVERVIEW

We present an overview of Sage, by describing each of the aforementioned stages in terms of user flow and back-end algorithms.

Data Collection and Preprocessing. While much research into recommender systems is focused on numeric ratings, the common case in most real-world systems is non-numeric implicit data. Under this setting, only positive observations are available – hence the name *one-class* data [6]. Sage recommendations are based on implicit one-class data, e.g., purchase history, movies watched, etc. We do not assume the presence of numeric ratings or even binary ratings.

Data collection requires uploading an items catalog. The catalog may also include item meta-data features. The training data can be either dynamically collected from a website using embedded scripts or uploaded offline as a file. The preprocessing consists of data cleaning, in which non-informative data rows are pruned, e.g., items that were purchased only few times and alone. If item meta-data is included in the catalog, we also employ a novel feature selection algorithm in order to choose a subset of meta-data features which are most informative with regard to collaborative filtering. Our feature selection algorithm is described in [8].

Offline Modeling. Recommender systems based on Matrix Factorization (MF) models have repeatedly demonstrated better accuracy than other methods, such as nearest neighbor models and restricted Boltzmann machines [1, 4]. How-

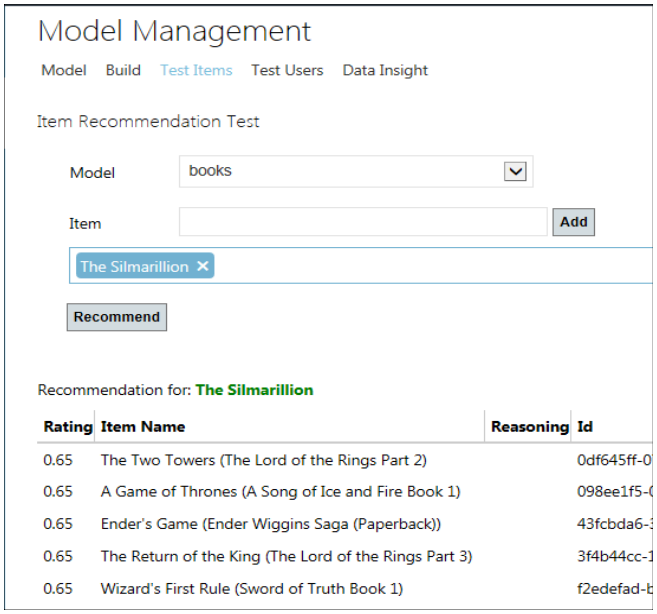

Figure 2: Qualitative evaluation dashboard (with books recommendations)

ever, MF models based on one-class data are relatively new. Sage uses a novel in-house probabilistic Bayesian algorithm that was recently published in [7]. Based on this model, Sage computes probabilities p_{mn} – the probability that user m will consume item n. In this process, Sage also employs items meta-data in a similar fashion to [5]. At the modeling stage, we also extract item-to-item relations (similarities) for common co-occurring item pairs.

After the training stage, Sage performs an evaluation process. A quantitative evaluation is computed in terms of *Mean Percentile Rank* (MPR), a relatively new metric in studies of implicit feedback recommenders [9]. Qualitative evaluation is also provided in the form of an interactive dashboard. The dashboard allows users to choose any subset of items, and generate appropriate recommendations as well as explore item-to-item relations (see Figure 2).

Online Module. The online module enables retrieving recommendations either for existing users or for new users (i.e., users not present in the training data). In the former case, recommendations are based on past interactions collected during the data collection stage. In the latter case, Sage quickly trains the model, on the fly, to include new users based on a short list of items that the new user is currently considering (click log, shopping cart, etc.). Similarly, Sage can update existing users online, based on new observations.

The Sage APIs. As discussed above, an API is provided to allow simple embedding of the recommender service in a live website. Figure 3 shows an example of the Sage API. The reader will notice methods to build a model and to retrieve item-to-item relations and personalized recommendations (*I2IRecommendation* and *U2I Recommendation*, respectively). Automatic data collection is performed by embedding a Javascript snippet on each item page. Usage information about the item (purchases, views, etc.) is collected and uploaded directly to Sage.

```
//-------------
// Sage API
//-------------
Gate.Login(http, userName, password);
var model = Models.CreateModel(http, modelName);
Models.UploadCatalogFileForAModel(http, model.Id, catalogPath);
Models.UploadUsageFileForAModel(http, model.Id, usagePath);
Models.ListModels(http);
Build.BuildModel(http, model);
Build.ReportModelsProgress(http);
Models.DeleteModel(http, model.Id);

Recommend.I2IRecommendation(http, modelId, itemId);
Recommend.I2IRecommendationWithListOfItems(http, modelId, history);
Recommend.U2IRecommendation(http, modelId, userId);
Recommend.U2IRecommendationWithHistory(http, modelId, userId, history);
```

Figure 3: Using Sage API in C# code: upload data, build model and retrieve recommendations.

3. FUTURE WORK

Research and development at Sage are planned to be extended in several directions. We wish to implement additional recommendation algorithms. In some scenarios, we believe that users may benefit from other algorithms, such as non-probabilistic matrix factorization, neighborhood-based recommender systems, content-based filtering, and hybrid models [2].

Real world models are best evaluated against real-world results. We also plan to develop an experimentation framework [3] to evaluate different models based on metrics such as Click-Through Rate (CTR), or conversion rate, rather than just offline metrics such as MPR.

4. REFERENCES

[1] Lessons from the Netflix prize challenge. R. M. Bell and Y. Koren. SIGKDD Explor. Newsl. 2007.

[2] Hybrid Recommender Systems: Survey and Experiments. Robin D. Burke. In User Model. User-Adapt. Interaction, 2002.

[3] Online Experiments: Practical Lessons. R. Kohavi, R. Longbotham. T. Walker. Online Experiments: Practical Lessons, 2010.

[4] The Yahoo! Music Dataset and KDD-Cup'11. G. Dror, N. Koenigstein, Y. Koren, M. Weimer. Journal Of Machine Learning Research, 2012.

[5] Xbox Movies Recommendations: Variational Bayes Matrix Factorization with Embedded Feature Selection. N. Koenigstein, U. Paquet. In ACM RecSys 2013.

[6] One-Class Collaborative Filtering. R. Pan, Y. Zhou, B. Cao, N.N. Liu, R. Lukose, M. Scholz, Q. Yang. IEEE International Conference on Data Mining, 2008.

[7] One-class Collaborative Filtering with Random Graphs. U. Paquet, N. Koenigstein. In Proc. WWW 2013, 999–1008.

[8] Selecting Content-Based Features for Collaborative Filtering Recommenders. R. Ronen, N. Koenigstein, E. Ziklik and N. Nice. In ACM RecSys 2013.

[9] Training and testing of recommender systems on data missing not at random. H. Steck. In KDD, 2010.