PrivateRecSys

System Architecture

25th July, 2021

Introduction

This deliverable presents the architecture of the PrivateRecsys system to facilitate its development and investigates further application, through its integration into Searx.

The document starts with a short introduction to privacy preserving recommender systems and describes the architecture of thePrivateRecsys system. This step facilitates the incremental development of the project and provides guidance to anyone wanting to modify, improve or extend the PrivateRecsys system in the future.

Further, this document presents the architecture of Searx - the popular open-source meta-search engine, which this project aims to utilize and be applied on. Ways to integrate PrivacyRecsys into Searx are identified and discussed.

This guide can be useful to anyone interested in extending the PrivateRecSys project or building other integrations for the Searx system.

Privacy Preserving Recommender Systems

What are recommender systems?

Recommender systems are systems that make interesting suggestion to the users related to items such as news and articles, products or even search results in timely manner and in the right context.

Traditionally recommendation algorithms are broadly divided into three categories: Collaborative Filtering , Content Filtering, and Hybrid.

1. Content filtering: These algorithms deliver recommendations to a user, by comparing the users' extracted interests with the item's characteristics. If the user has previously shared experiences that denote an interest of the item (and therefore the user and the item have some similarity) the item will be recommended to the user.

2. Collaborative filtering: These algorithms work without prior information of either user or item. Collaborative filtering algorithms build user profiles to then compare them to profiles of other users. When identifying that user A has certain similarity with user B, the algorithm will recommend items user B has consumed in the past to user A. Collaborative filtering via Matrix factorization has become the default choice to build recommendation systems.

3. Hybrid filtering: These algorithms are a combination of the above two algorithms. It encompasses methodologies to take pros from both the above techniques to build better recommendation systems.

Another suite of techniques that is interesting in the domain of ranking/recommendation/search are called Learning to Rank methods. In Rank methods, a loss function is built based on the propensity of a user interested in an article and then rank it accordingly. There is a pair-wise learn to rank model, which optimizes the number of inversions between pairs. Also, there is list-wise learn to rank, which optimizes either directly the similarity between actual rank and predicted rank or the metrics directly, such as Mean average precision (MAP).

Why is privacy needed in recommendations?

Privacy has become a troublesome issue for both the service provider and end user, and it has received interest from the recommender systems community. There are also certain advantages to introducing privacy in recommendations/search. Introducing privacy allows to leverage better

metadata, which the user doesn't share, such as app information on the phone, location, etc. Better adoptions of the machine learning models can be produced that fully respect users' privacy in data- sensitive domains such as healthcare, financial services, when the user is comfortable in using the system without the resistances of the data being shared or looked at by other people.

Privacy Preserving Recommendation System

There are several ways we can build a recommendation system with privacy:

- **Learn to Rank Pointwise**: We can build a neural network to predict user's clicking behavior. Essential given:
    - P(U,I) = 1 (if click) else 0
    - We can rank items based on the probability score from the model.
- **Collaborative Filtering:** Matrix factorization models can be used to store the user's interactions on the device side, in the form of a user profile. By storing the user profile on the client side, as well as the recommendation calculations, we achieve privacy preservation. The downside can be increased need in resources such as memory, to store the data, and computational power to perform calculations.

In this project, we focus on developing a federated collaborative filtering model via matrix factorization. the

Matrix Factorization

Matrix Factorization (Low rank factorization methods) decomposes the matrix into the product of two lower dimensional rectangular matrices and are typically applied into an interaction matrix. Interaction matrix is essentially a matrix where the rows are your users and columns are the items and the cell is the interaction (in some cases ratings, clicks etc.) associated with the item by the user. The reason to apply matrix factorization is to reduce sparsity in the original interaction matrix (In most cases this is above 90% of the actual matrix). The resultant lower dimensional matrices are much richer thus we can obtain ratings for missing interactions (ratings) by simply taking the dot product of the row (specific user row 'x_u') and column (item row 'y_i') of the lower dimensional matrix example and do a dot product.

$$\hat{r}_{ui} = \mathbf{x}_u^\intercal \cdot \mathbf{y}_i = \sum_k x_{uk} y_{ki}$$

Predicted rating

To obtain the resulting matrix, we can minimize the loss function of the actual ratings (r) and predicted ratings. In the following equation we have also added the regularization terms.

$$L = \sum_{u,i \in S} (r_{ui} - \mathbf{x}_u^\intercal \cdot \mathbf{y}_i)^2 + \lambda_x \sum_u \|\mathbf{x}_u\|^2 + \lambda_y \sum_u \|\mathbf{y}_i\|^2$$

Loss function with regularization

The objective function is a non-convex in this formulation. Gradient descent can be used as an approximate approach here, but it's slow and consists of lots of iterations.

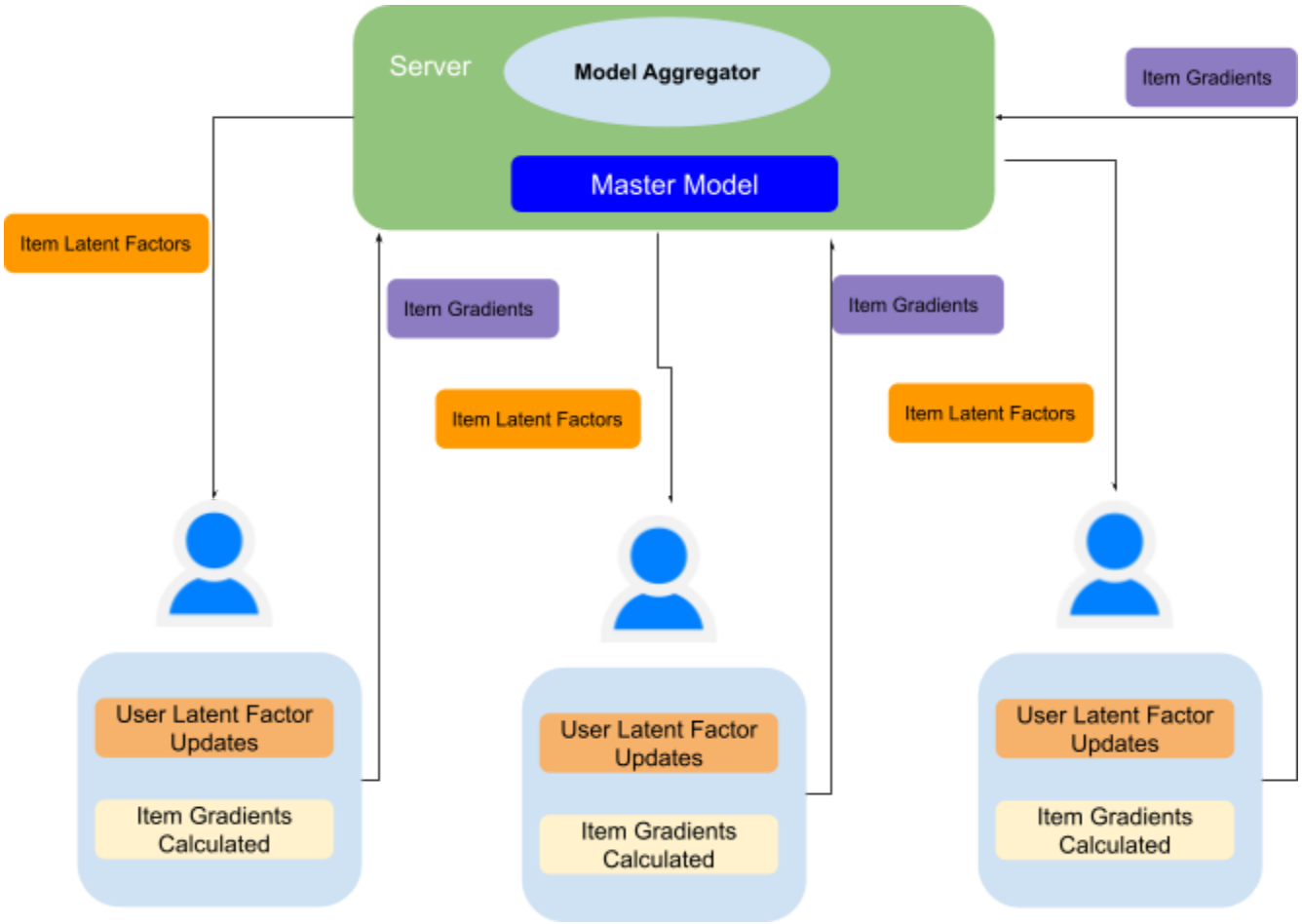
However if we fix a set of variables X and treat it as constants, then the objective is a convex function of Y and vice versa. Our approach will follow this route, we will fix Y and optimize X, then fix X and optimize Y, and repeat until convergence. This algorithm is called Alternating Least Square (ALS).

Federated Collaborative Filtering (Federated ALS)

To make the ALS algorithm trained in federated fashion and be focused on preserving user's privacy, we proceed into modifications in the way we train the model.

1. All the item factors vectors are updated on the central server and then distributed to each client (device/user).
2. The user's latent vectors are trained on the device using the user's own data with item latent vectors.
3. The gradients for each item vector are calculated and propagated back to the central server, where it's aggregated and item vectors updated.

The following model demonstrates how the privacy preserving recommender system will work showcasing the Master model with the Model Aggregator on the server, and each client holding their profiles with their personal Item Gradients calculated and the User Latent Factor Updates. The server will send each of the clients the derived Item Latent Factors and the User will then send back to the server the Item Gradients.



Searx System Architecture and Integration investigation.

Searx System Introduction

Searx is a free open-source internet metasearch engine available under the GNU Affero General Public License version 3. It's open-source nature and extendability, make it an ideal platform for further development and experimentation and this is why it was selected as the application platform for this project.

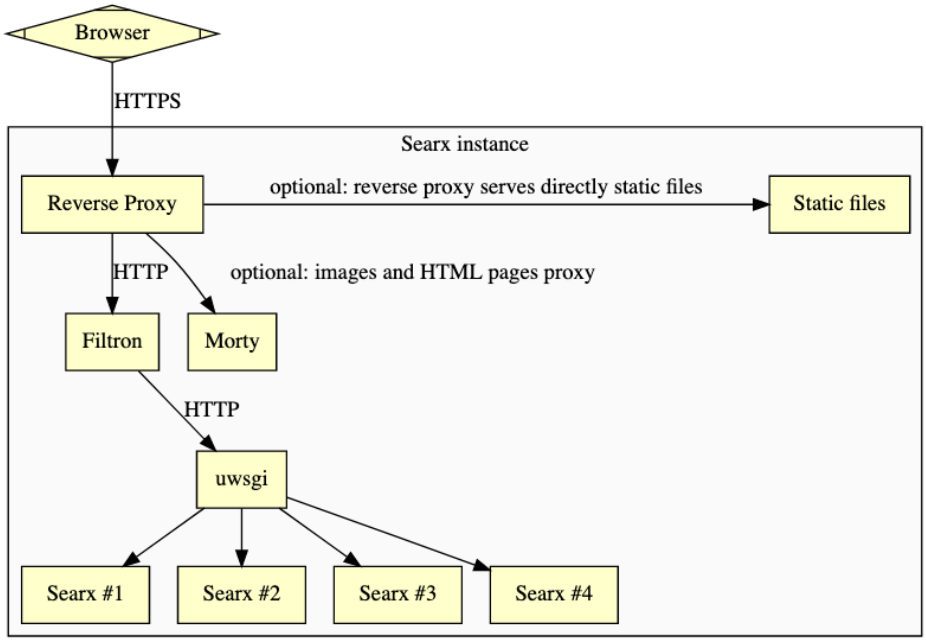
Searx was developed with the aim of protecting the privacy of its users. Searx does not track or profile users and does not share users' IP addresses or search history with any of the more than 70 search services which it uses to produce results. Tracking cookies served by the search engines are blocked, preventing user-profiling-based results modification.By default, Searx queries are submitted via HTTP POST to prevent users' query keywords from appearing or staying in web server logs.

Searx facilitates general search, specific domain search and also supports cached and proxy links. Each search result is given as a direct link to the respective site, rather than a tracked redirect link as used by Google. In addition, when available, these direct links are accompanied by cached and/or proxied links that allow viewing results pages without actually visiting the sites in question. The cached links point to saved versions of a page on the Wayback Machine, while the proxied links allow viewing the current live page via a Searx-based web proxy. In addition to the general search, the engine also features tabs to search within specific domains: files, images, IT, maps, music, news, science, social media, and videos.

A privacy-preserving recommender system could produce suggestions for either of search specific domains, as well as filtering and providing more relevant generic queries. The focus of this project is on filtering results using global/local/ privacy preserving search as well as enabling users to receive product recommendations - products can include purchasable products but also items such as news or music.

Searx System Architecture

The following diagram demonstrates the system architecture of a typical public searx instance. The system is made out of different components including :



Searx as shown in the diagram, contains the following components:

- Reverse Proxy: Apache & nginx
- Filtron: Filtron is just a middleware between your web server and searx, to limit the requests processed by searx and prevent abusing external services.
- Morty: Web content sanitizer proxy as a service. Morty rewrites web pages to exclude malicious HTML tags and attributes. It also replaces external resource references to prevent third party information leaks.
- uWSGI: which provides a full stack for building hosting services and supports the searx instances.

The modification Search can overall be achieved in two ways: a) by running private instances of Searx, either locally, or in public servers, or b) by using Meta-Search instances. Meta-Search

instances can forward the search query to a random public instance. A public API is available for Searx as well as Firefox search provider plugins.

The extension of the functionality of Searx can be accomplished by developing external plugins and registering them to the system.

Plugins

Plugins can extend or replace functionality of various components of searx. They are developed as standard python modules implementing all the requirements of the standard plugins. Plugins can be enabled by adding them to settings.yml plugins section. Searx provides an indicative external plugin as an example here.

Plugins can be enabled by registering them in searx > plugin > __init__.py. Using plugins.register(name_of_python_file)

Entry points (hooks) for each plugin need to be defined. Searx currently supports only three hooks. The Pre search hook, which runs before the search request (pre_search), the Post search hook, that runs after the search request (post_search) and the Result hook which runs when a new result is added to the result list. (on_result).

Plugins can be used both to alter the UI but also filter results according to different streams.

External Search Engines

External Search engines can be used to alter the results of Searx and produce the desirable PrivateRecsys recommendations. Searx supports various external search engines using Adapters. Searx does not have a general search API which can be used for every search engine. Therefore, an adapter has to be built between Searx and any other external search engines. Adapters can be developed and be stored under the folder git://searx/engines.

Both plugins and External Search engines , as well as other external services can be used to enhance Searx functionality and use it as a case study for PrivateRecSys.

Conclusion

In this deliverable the architecture of the PrivateRecsys system is presented. The deliverable also investigates the application of the PrivateRecsys to Searx.

This document will be updated after the development of the system, to include the components analysis as well as a sequence diagram. This will provide a better understanding of how the system works and how one can extend and modify it.