**SOR – Social Recommender**

**Table of content**

**1. Start to know SOR.**

**2. Objectives and constraints.**

**3. Register**

**4. Sign in**

**5. Homepage**

**6. Resources:**

**7. Social networks: Facebook**

**8. FOAF graph**

**8.1 Creation of FOAF graph**

**8.2 Storage of FOAF graph**

**8.3 Usage of FOAF graph**

**9. API specifications**

**9.1 RAML file**

**9.2 Link to RAML file**

**10. Results - limited display of results**

**11. Caching**

**12. User preferences**

**13. Conclusion**

**14. Bibliography**

1. **Start to know SOR.**

Social recommender is an application that is able to recommend certain people/events/things of interest according to a given FOAF graph build with information from a social network (for now Facebook, but later on it has the possibility to be extended with more information from other social networks such as LinkedIn, Twitter etc.) and external resources. The information from the social networks is about the client's features, interests, likes/dislikes, events/things, heard/liked music, artists, profession, knowledge, places he/she visited, played games, persons he/she knows and interacts with.

1. **Objectives and constraints.**

Create an application that recommends (randomly or on request) to the user certain people/events/things of interesting according to social networks (Facebook).

Application constraints:

* The application should be a responsive application
* Service oriented application
* Restful API
* An application built on the existing social and semantic Web technologies
* The code-source and specific content must be available under the terms of the open source licenses
* To be discussed

1. **Register**

First step to create an user account is to **register**. The registration requires filling in a form with personal information about the client so that when the user first checks the application, some preferences would already be displayed on the start page.

Note: The email and password are unique for every user.

1. **Sign in**

Using the information from the registration step, like email and a password, the user can sign into a defined account already containing random recommendations based on details provided in the registration form.

1. **Homepage**

Homepage is the staring page of the application and contains either results of random recommendations based on a social network (Facebook) and saved preferences – likes/dislikes (if any), or the results of a custom, user-defined query, also considering user’s feedback on previous results.

**6.** **Resources**:

Resources are used to get accurate recommendations for the user. We use multiple open data sources in RDF format and their corresponding ontologies.

Here are some examples of resources that could be used for this application:

* **Dbpedia** *is a crowd-sourced community effort to extract structured information from Wikipedia and make this information available on the Web.*
  + <http://wiki.dbpedia.org/>
* **GeoNames.org - t***he geographical database covers all countries and contains over eight million place names that are available for download free of charge.*
  + http://www.geonames.org/
* **Linked Data** *is about using the Web to connect related data that wasn't previously linked, or using the Web to lower the barriers to linking data currently linked using other methods. More specifically, Wikipedia defines Linked Data as "a term used to describe a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF.*
  + http://linkeddata.org/
* **Schema.org** *is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond.*
* **Freebase** *was a large collaborative knowledge base consisting of data composed mainly by its community members.*
  + https://www.freebase.com/
* **BabelNet** *is a multilingual lexicalized semantic network and ontology.*
  + http://babelnet.org/
* **Wikidata** *is a collaboratively edited knowledge base*.
  + https://www.wikidata.org/wiki/Wikidata:Main\_Page

and so on.

**Note**: This list of data sources is not final and the definitions of the resources are taken from the main sites.

**7. Social networks: Facebook**

The FOAF graph will be constructed based on information from Facebook. The user must be connected to Facebook in order to create the graph and then be provided with the random or requested recommendations.

8. **FOAF graph**

FOAF graph (friend of a friend) is a RDF file that stores personal information taken from a social network (Facebook).

Personal information means information from Facebook profile, including client's friends, likes, events, things, professions, music, artists, work places, educational institutes, domains of work and study, stores, electronics, clothes and so on.

**8.1** **Creation of FOAF graph**

A FOAF graph is created first when the client connects to a social network, in a RDF format (.xml/.ttl). The client has the possibility to update the content of the graph every time he/she requests so.

Note: Time limit of updating the graph can be changed if necessary.

**8.2** **Storage of FOAF graph**

FOAF graph will be loaded on the server side every time it is created/updated. There will be a RDF graph or more for each user that has an account for this application.

**8.3 Usage of FOAF graph**

The graph is used for:

* random generated recommendations displayed on homepage
* on user request generated recommendations.

**9. API specifications**

RAML file is used to describe API specifications

**9.1 RAML file**

RAML file describes the API specifications of the application. The API is structured in four main modules:

* **Randomly generated recommandations**

Based on users’s FOAF graph, the app will first extract similarities – most common occurrences of things (places, movies), most popular events. It will also take into account user’s geographical location and current town as listed on Facebook. Considering this data, it will search for events nearby, similar movies, people who went to the same school and other things. Finally, the user will see query results filtered by type on his home page, or he can request the list by calling the following endpoints: /recommendations?item\_type=[TYPE\_OF\_ITEM], where TYPE\_OF\_ITEM is one of: person, event, movie, book, place, educational\_institute. A drop-down with the five classes of recommendations will be available, serving as a filter where the client can chose the type of recommendations to see.

* **Graph content**

The user can see the content of his generated FOAF graph, sorted by relevance and filterd by item type, by calling the following uri:

/graph?item\_type= [TYPE\_OF\_ITEM], where TYPE\_OF\_ITEM is one of: person, event, movie, book, place, educational\_institute.

* **Search module**

The user will be able to look for stuff by defining his own criteria in five different forms for each type of resource. Some examples – the client wants to get all books of Agatha Christie first published after 1940; the client wants to get all universities with over 5000 students and at least 2 Nobel Prize laureates.

All search features will be available by querying the following endpoint:

/search/[TYPE\_OF\_ITEM], with TYPE\_OF\_ITEM one of: person, event, movie, book, place, educational\_institute.

* **Saved items**

The user will have the possibility to give positive/negative feedback for each query result, by clicking like/disliked. The item specified will be stored into database. Disliked items will not be taken into account on the next query (or, from the client point of view, on the next set of recommendations, either random or custom). Saved items can be filtered by type or grouped into user-defined lists with suggestive names. An items can be on many lists or on none. For reasons of time and space complexity, only at most three-months old saved items will be considered on queries.

By calling the endpoint /saved-items?item\_type=[TYPE\_OF\_ITEM], the user will get the list of all items. For lists, a call to /lists/[list\_id] is required.

**9.2** **Link to RAML file**

**10.** **Results - limited display of results**

Recommended items are query results from a set of RDF graphs, used by our app and listed in resource section. [link] On each query, the user will also send a format parameter, which can be application/json, html or RDF.

Results are displayed in Homepage, containing the name of the results and other information, according to the type of result.

The constraints of the number of items displayed are:

* minimum number of results: 5
* maximum number of results: 50
* default: 10

**11. Caching**

Queries send by the user are saved on server side, in our database. This way, they will be available to the client from any computer, anytime.

**12. User preferences**

A list of preferences will be saved for the user profile when he/she indicates the results liked by the user.

Beside this list, SOR also creates a blacklist of results that should not be included in the displayed results. In this way SOR becomes more accurate each time it is used.

Two checks will be available for each result one with ‘+’ meaning to include this result in the user profile, and one with ’ –**‘**, meaning to exclude this result when receiving a recommendation.

Preferences are by default kept from the first time the user signed in his account, but there is a checklist available where the user can select the period of time the preferences were added.

**13. Conclusion**

Social Recommender is a web application that user social networks (Facebook) and other external resources to recommend the user certain people, events, things, movies, books and so on.

**14. Bibliography**

DBpedia: <http://wiki.dbpedia.org/>

FOAF: <http://www.foaf-project.org/>

RAML: <http://raml.org/>

Facebook: <https://www.facebook.com/>

Security schemas: <https://api.yaas.io/patterns/v1/security-schema-basic.yaml>

Schema.org: <https://schema.org/>

Wikipedia: <https://wikipedia.com/>

Blog: <https://socialrecommender.wordpress.com/about/>

Web project page: <http://profs.info.uaic.ro/~busaco/teach/courses/wade/web-projects.html>