An Architecture for smart semantic Recommender Applications

Andreas Lommatzsch, Till Plumbaum, Sahin Albayrak

Competence Center Information Retrieval and Machine Learning
DAI-Labor, TU Berlin
Ernst-Reuter-Platz 7
D-10587 Berlin
{andreas.lommatzsch,till.plumbaum,sahin.albayrak}@dai-labor.de

Abstract: With the growing availability of semantic datasets, the processing of such datasets becomes the focus of interest. In this paper, we introduce a new architecture that supports the aggregation of different types of semantic data and provides components for deriving recommendations and predicting relevant relationships between dataset entities. The developed system supports different types of data sources (e.g. databases, semantic networks) and enables the efficient processing of large semantic datasets with several different semantic relationship types. We discuss the presented architecture and describe an implemented application for the entertainment domain. Our evaluation shows that the architecture provides a powerful and flexible basis for building personalized semantic recommender systems.

1 Motivation

The amount of available semantic data grows rapidly. Semantic data sources (such as DBpedia[1] or Freebase[2]) provide huge collections of structured data. These semantic datasets store knowledge as a network of nodes and edges. The nodes represent entities (e.g. persons or items) and the (labeled, weighted) edges the relationships between these entities. The representation of data as semantic networks avoids problems of text based data collections, such as spelling mistakes, homonyms, synonyms or ambiguous terms. Semantic networks are language independent, and easily extendable by new entities and relationships.

Knowledge stored in semantic datasets can be deployed in many application scenarios. A very active research area is the use of semantic data for information retrieval and recommender systems. The challenges for the design of recommender applications based on semantic datasets are:

[1] http://dbpedia.org
[2] http://www.freebase.com

The efficient handling of large semantic datasets: Semantic datasets usually consist of a large number of nodes and edges. Nodes as well as edges might be annotated with meta-data defining weights or type information. The semantic data must be stored in a format enabling an efficient data access. The data format should consider the requirements of relevant recommender algorithms and potentially extension for new applications.

Support for different edge types (semantics): In general, semantic networks contain several different semantic edge types. A system for handling semantic networks must be able to cope with different semantic relationship types and support rules for combining edges of different types within a path.

Support for various network models: For an efficient processing of large networks, the systems should support different network models to reduce the complexity of large semantic graphs. These models restrict the network to the most relevant components ("latent concepts") and reduce the noise existing in real-world data sets. A universal framework for processing semantic data should support several different types of models, so that dependent from the respective scenario the best matching models are selected.

High quality prediction and explanations: The framework for handling semantic datasets should provide different recommender algorithms allowing a scenario and context sensitive selection of the most appropriate algorithms. The deployment of adequate recommender algorithms ensures reliable high quality recommendations. For improving the acceptance of recommendations the system should provide explanations. Explanations help the user to understand why an entity has been recommended and to which extent the computed entities match the user's preferences.

Open for new data sources and algorithms: Semantic datasets become relevant in a growing number of domains. Thus, a framework for the processing of semantic data must be open for new datasets. Moreover, the framework should support the integration of new dataset models and recommender algorithms. This ensures that new application domains can be managed by the system.

In this paper, we introduce our framework ("Semantic Engine") that provides the basis for the processing of large semantic data collections and that integrates various methods for computing recommendations.

The remaining of this paper is structured as follows. Section 2 gives an overview to current recommender algorithms and methods for processing semantic data. Section 3 introduces our framework and explains the components and implemented methods in detail. Subsequently we present an implemented recommender system based on our framework. Finally a conclusion and an outlook to future work are given.

2 Related Work: Existing Systems

Current systems for processing semantic data focus on recommender systems in social networks. Typical use cases are link prediction in social networks (user-user-relations) and product recommender (user-item relations). The datasets are handled as a large network representing users and entities as nodes and the relationship between the entities as weighted edges. The most frequently applied approaches for computing recommendations in semantic networks are content based filtering and collaborative filtering.

2.1 Content based Filtering

Content based filtering algorithms compute the relevance of entities based on features ("content") of entities [La95, AWL$^+$07]. The most important terms are extracted from the textual description of entities and used for calculating the similarity between the entities. For computing recommendations for a given entity the most similar entities (e.g. based on a weighted term overlap) are determined.

The problem of content based filtering techniques is that additional effort is needed for handling multi-lingual content and for taking into account homonyms, synonyms, and anaphors. Commonly used approaches calculating the term weights based on term-frequency statistics do not compute reliable prediction results for incorrectly coded texts and documents containing spelling mistakes or ambiguous terms.

2.2 Collaborative Filtering

Collaborative filtering algorithms compute the relevance of entities based on the similarity of users and items [SKD$^+$06, HKB$^+$99]. The underlying assumption is that users who agreed in the past tend to agree again in the future. Thus, collaborative filtering suggests items *liked* by users with similar interests or items similar to items *liked* in the past. The features of the recommended items are not taken into account. Collaborative filtering avoids problems of multi-lingual texts and ambiguous words. The disadvantage of collaborative filtering is that predictions can only be computed for entities having several edges to other nodes. That is why collaborative filtering provides less precise results for new users (also known as the "cold start problem") and for datasets containing only a small number of edges compared to the number of entities ("sparsity problem"). Moreover, most collaborative filtering systems can only handle one type of relationship (usually "is related" or "liked"). Thus, additional effort is needed for handling dataset containing several different relationship types.

2.3 Problems with existing Approaches

Current recommender approaches are usually tailored for one scenario. The parameter settings are often chosen by an expert to meet the specific requirements of that scenario [AZ05]. Furthermore most recommender systems base on bipartite datasets and cannot handle large semantic networks with differently labeled edges. Existing hybrid recommenders often statically combine a content-based and a collaborative approach based on expert defined rules. Thus, for every new application the recommender framework must be manually adapted.

2.4 Summary and Conclusion

Most of existing frameworks for processing semantic data do not fulfill the requirements according to openness and flexibility. Many systems are designed for a single scenario but lack extensibility and the support of several semantic relationship types. In the following section we introduce our Semantic Engine framework that provides a comprehensive collection of components for processing large semantic data collections and for building smart recommender applications.

3 Approach and System Architecture

In this section we introduce the developed system architecture and discuss the implemented components and algorithms in detail. Our framework is designed to work with large semantic datasets internally managed as semantic networks, consisting of nodes (representing entities e.g. users, interests, artists, etc.) and labeled, weighted edges. For handling large real-world networks the framework supports datasets with multiple semantic relationship types and uses a data representation optimized for sparse networks.

The developed architecture consists of four main components, shown in Figure 1.

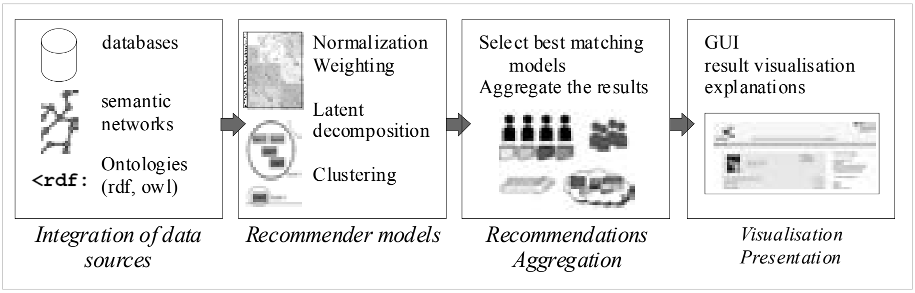


Figure 1: The semantic engine components.

The leftmost component connects the recommender with different types of data sources. Data can be imported from ontologies or triple stores (e.g. Virtuoso[3]). Moreover different method are supported for importing data from "traditional" sources (e.g. from relational databases) or structured text files.

The imported data are preprocessed to extract knowledge needed for computing recommendations and to speedup the calculation. The framework provides methods for data normalization, defining weights of nodes and edges and for clustering. Additionally, components for noise reduction and the extraction the most relevant information are provided (e.g. based on latent matrix decomposition).

The third component defines rules and models for computing recommendations. The framework supports memory based recommenders as well as model based recommenders. A meta-recommender selects the most appropriate models and aggregates the results computed by different recommenders based on the respective scenario.

The rightmost component in Figure 1 shows the component that provides functions for visualizing the results. The component generates explanations for computed recommendations and provides an interface to external services.

The next paragraphs explain the components and implemented algorithms of our framework in detail. We focus on the recommender framework and the machine learning algorithms for optimizing the parameter settings.

3.1 Representing semantic Datasets as Graphs

In our framework the semantic data are stored as a graph. Figure 2 shows an example for a music dataset containing the entity types "Artist and Band", "Genre", "Album", "Track" and "User". The edges define the semantic relationships between the entity sets. E.g., the relationship set "AlbumRelease" defines which albums have been released by an artist or a band; the relationship "loved Artists" provides information which user "liked" which artists.

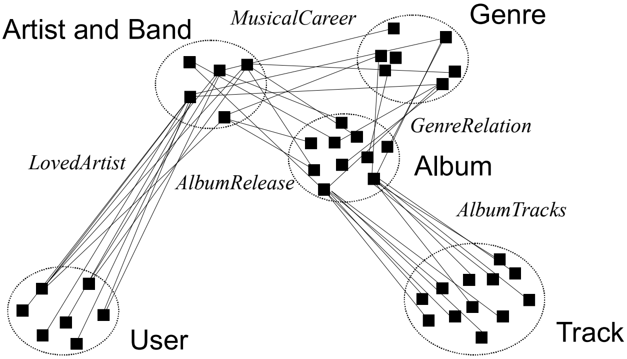


Figure 2: A graph representing a semantic dataset for the music domain.

[3] http://virtuoso.openlinksw.com

3.2 Normalizations and relative Weights

For computing recommendations based on a graph, it is necessary to assign appropriate weights for each edge. The weights must consider the semantic meaning ("label") and the relevance of the edges. Usually, edge weights are scaled to values between *0* and *1*. The developed framework supports additive and multiplicative normalization, such as subtracting the overall minimal weight or dividing by the overall maximum edge weight.

In semantic datasets containing several different semantic relationships a weighting model for each edge type must be defined. The developed framework supports expert defined edge weights as well as machine learning algorithms, deriving the optimal weights based on a training dataset.

3.3 Learning a Network and a Prediction Model

We define a model for combining the edge weights of a path between two nodes. The model must consider the case of parallel edges and the case of an edge sequence. Figure 3 shows three approaches for defining an edge algebra. In most applications, for simplicity we apply the model of weighted paths.

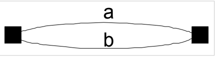
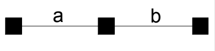
	Weighted Path	Resistance distance	Shortest Path
	a+b	(a*b)/(a+b)	min(a,b)
	a*b	a+b	a+b

Figure 3: Different edge algebras

Another important step for computing recommendations is to specify a prediction model that defines which properties an entity must fulfill to be relevant for a query. The most commonly used recommender model is the semantic similarity of an entity with the input entities. This model follows the idea that entities connected with short paths (having a high weight) to the input entities and entities for that several parallel paths exist are most relevant to the input entities. Alternative recommender models are triangle closing or number of common neighbors [LHK10].

3.4 Memory based Recommender

For computing recommendations, the framework supports memory based and model based recommender algorithms. Memory based recommender calculate paths in the original dataset. Starting from a set of given input entities (e.g. entities in the user profile), the potentially relevant paths through the semantic network are examined. The entities reachable from the input entities are ordered according to a semantic similarity rating. This rating is calculated based on the edge weights of the respective paths. For parallel edges/paths the ratings are summed up; for a sequence of edges the weights are multiplied and weighted by a discount factor (dependent on the path length).

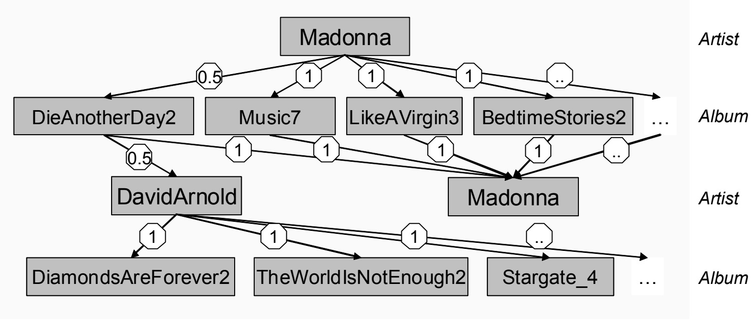


Figure 4: Search related entities in a semantic network (memory based recommender)

Figure 4 shows paths through a bipartite (artist-album) network. Starting from the artist "Madonna" semantically related entities are computed.

The advantage of memory based recommenders is that updates in the semantic network immediately effect the computed recommendations. Moreover, the paths from an input entity to a "recommended" entity can be presented to the user as a simple explanation [SNM08]. The disadvantage of memory based recommenders is that they provide not as precise results as model based recommenders on sparse networks. Another disadvantage is, that the computation of long paths is resource-intensive.

3.5 Model based Recommender

Real world datasets are often sparse, noisy and very large. Thus, building models based on these dataset enables an effective complexity reduction by restricting the spanned search space to the most relevant dimensions. Our framework deploys algorithms for computing low-rank approximations of complex datasets [SKK+02, KL09] and for clustering similar entities [JMF99]. The clustering component supports the integration of additional expert knowledge to improve the computed network model. Figure 5 visualizes two methods for building a network model.

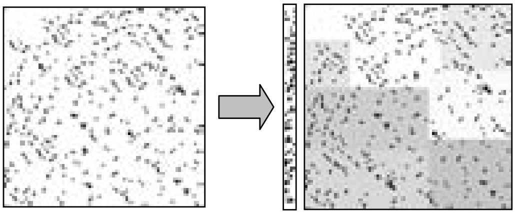


Figure 5: The figure schematically visualizes the complexity reduction for a graph, represented as a sparse matrix. Supported methods for building network models are the dimensionality reduction based on a matrix-decomposition and on clustering.

The computation of related items in a semantic network containing clustered entities is shown in Figure 6. In general, path based search algorithms can be applied. Due to the reduced network complexity longer paths through the network can be computed efficiently requiring only a reduced amount of recourses. The disadvantage of model based recommenders is that additional effort is needed when the dataset changes. Another disadvantages is, that network models are often difficult to understand. Thus, it is not easy to provide human readable explanations with model-based recommenders.

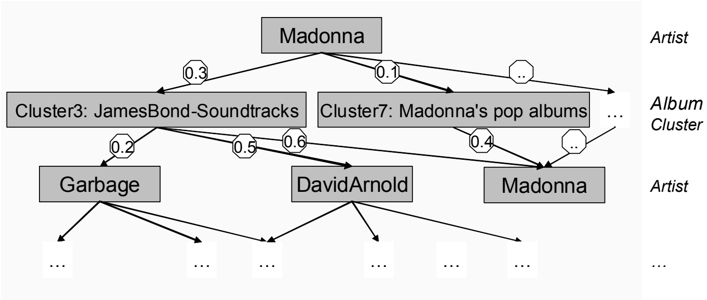


Figure 6: The figure visualizes the search for related entities in a semantic network containing a clustered entity set.

3.6 Meta Recommender

A framework for processing semantic data should be able to integrate several different recommender algorithms and models [Po06]. Our framework provides a meta-recommender that selects the most appropriate recommender algorithms based on the respective scenario and the query. Results from the chosen recommenders are aggregated in a unified result list. The meta-recommender strategy can ensure the "diversity" of the recommended entities (meaning the entities have been calculated based on different models) or on the consensus of different recommender models (e.g. computed based on the aggregation strategies CombSUM or CombMIN [Lee97]).

3.7 Summary and Evaluation

In the previous section we introduced the main components of our framework for processing of semantic data and for computing recommendations in detail. Additional components for optimizing parameters, evaluating the quality of the recommender and for providing recommendation results as a web service exist.

The evaluation of our approach on various semantic dataset has shown that the framework is able to process many different types of semantic datasets and can provide high quality recommendation results. A recommender application for the entertainment domain is presented in the next section.

4 A Music Recommender based on encyclopedic Data

The Project SERUM[4] connects unstructured news with large semantic entertainment datasets. The key functionality is a recommender that suggests relevant entities (artists, albums, tracks) and news articles for potentially relevant artists. The basis for the recommender is an encyclopedic data set, retrieved from Freebase[5]. The semantic data set used in the project SERUM is shown in Figure 2. A screenshot of the implemented web application shows Figure 7.

The recommendations are computed based on the framework explained in Section 3. The user interface is implemented as a web application. The current application computes recommendations for explicit preferences entered by the user. The support of implicit feedback is planned for the near future.

Figure 7: A screenshot of the start page from the SERUM web application.

The first experiences with the SERUM web application show that based on the encyclopedic dataset and the developed framework highly relevant recommendation are computed. Due to the provided explanations the users understand, why entities have been recommended and to which extent they are semantically related to the user preferences. The size of the used encyclopedic datasets (≈2.0 Mio nodes and ≈1.6 Mio edges) allows computing recommendation even for only regionally known artist.

Unfortunately, the encyclopedic dataset does not provide ratings or information about the popularity of artists or albums. Thus, we plan to enrich the dataset with rating data. This will allow us to ensure that a recommended artist is not only semantically related, but also matches the user's individual preference.

[4] an acronym for "Semantische Empfehlungen basierend auf großen, unstrukturierten Datenmengen": http://www.dai-labor.de/irml/serum/. SERUM is a joint project between the TU Berlin and the Neofonie GmbH.

[5] http:// www.freebase.com

5 Conclusion and Outlook to future Work

The implemented framework allows the efficient processing of large semantic datasets (due to the different supported recommender models) and the creation of adaptive recommender systems. The available components support several different recommender algorithms and graph models. This enables the context aware selection of the most appropriate recommender algorithms. In contrast to many existing frameworks we support different semantic edges types within a semantic dataset and provide a meta-recommender that allows us to combine recommender results computed based on different recommender models.

The application SERUM showed that based on of our framework high quality recommender systems can be implemented. In future work we will focus on improving the support of unstructured data. Therefore, an additional component for Named Entity Recognition and Name Entity Disambiguation will be integrated in our framework.

Bibliography

[La95] Lang, K.: NewsWeeder: Learning to filter Netnews. In Proc. Int. Conf. on Machine Learning, San Mateo, CA, 1995.

[AWL+07] Albayrak, S; Wollny, S; Lommatzsch, A., Milosevic, D.: Agent Technology for Personalized Information Filtering: The PIA System. In Scalable Computing: Practice and Experience, vol. 8, Elsevier, 2007.

[SKD+06] Sahoo, N.; Krishnan, R.; Duncan, G.; Callan, J.: Collaborative filtering with multicomponent rating for recommender systems. In Proc. Workshop on Information Technologies and Systems, Milwaukee, WI, 2006.

[HKB+99] Herlocker, J.; Konstan, J.; Borchers, A.; Riedl, J.: An algorithmic framework for performing collaborative filtering. In Proc. Int. Conf. on Research and Development in Information Retrieval, New York, NY, 1999.

[AZ05] Adomavicius, G.; Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Trans. on Knowledge and Data Engineering, vol.17, Piscataway, NJ, 2005.

[LHK10] Leskovec, J.; Huttenlocher, D.; Kleinberg, J.: Predicting positive and negative links in online social networks. In Proc. of the 19th Int. conf. on WWW, Raleigh, NC, 2010.

[SNM08] Symeonidis, P. ; Nanopoulos, A.; Manolopoulos, Y.: Providing Justifications in Recommender Systems. In IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 38 (6), issn 1083-4427, New York, NY, 2008.

[SKK+02] Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J.: Incremental SVD-based algorithms for highly scalable recommender systems. In Proc. Int. Conf. on Computer and Information Technology, Dhaka, 2002.

[KL09] Kunegis, J.; Lommatzsch, A.: Learning spectral graph transformations for link prediction. In ICML '09: Proc. of the 26th Int. Conf. on Machine Learning, Montreal, 2009.

[JMF99] Jain, A. K.; Murty, M. N.; Flynn, P. J.: Data clustering: a review. In ACM Computing Surveys, vol. 31, issn 0360-0300, New York, NY, 1999.

[Po06] Polikar, R.: Ensemble based systems in decision making. In Circuits and Systems Magazine, vol. 6, IEEE, 2006.

[Lee97] Lee, J. H.: Analyses of multiple evidence combination. In SIGIR '97: Proc. of the 20th Int. ACM SIGIR conf. on Research and development in inf. retrieval, Philadelphia, PA, 1997.