Exploration of Semantic Data in Enterprise Applications

Cirrus Shakeri, Jascha Kanngiesser, Ralf Gueldemeister and Rei Kasai

SAP Labs, Palo Alto, California, USA
firstname.lastname@sap.com

Abstract. We have built a system that discovers new relationships between business objects in an enterprise application for customer support centers. The business objects represent customer service requests, knowledge-base articles about product problems and their resolutions, and information about the customer support personnel and their expertise. The new relationships discovered are of three types: similarity between the service requests, the knowledge articles related to a service request and the personnel with expertise related to a service request. The similar service requests, related knowledge articles and the people with expertise about the customer problem in hand are recommended by the system to the support personnel in order to help them resolve the customer issues faster.

Keywords: Semantic Discovery, Recommendation System.

1 Key Novelty of the System

The key novelty of the system is in shifting the paradigm in how people access and consume information in enterprise applications from search to recommendations. The main method for accessing information has been based on users explicitly searching for information using keywords. The new and emerging method for accessing and consuming information is based on recommending potentially relevant information to the users in the context of their activity.

We have demonstrated that the new paradigm is more effective in the domain of customer service where the support personnel work on resolving product issues or answering questions. The search paradigm puts the burden on the support agent to find the information that may help in resolving the issues faster. The user has to choose the right search keywords and iterate over the search results in order to find some potentially helpful information.

In the recommendation paradigm, the system proactively finds the potentially helpful information and makes it available to the support personnel if they chose to leverage it. This paradigm speeds up the process of resolving the customer issues by providing the most helpful information in a fraction of the time that a keyword-based search would take to find the same information.

2 The Users

The users of the system are customer support personnel. The system is especially helpful for junior personnel who have less experience or for situations where the products are too complex to be mastered by an individual.

The recommendations are generated in the form of similar service requests that have been successfully resolved in the past, knowledge-base articles that have relevant information and the name of other personnel who have expertise related to the specific problem. The overall usage scenario is shown in **Fig. 1**.

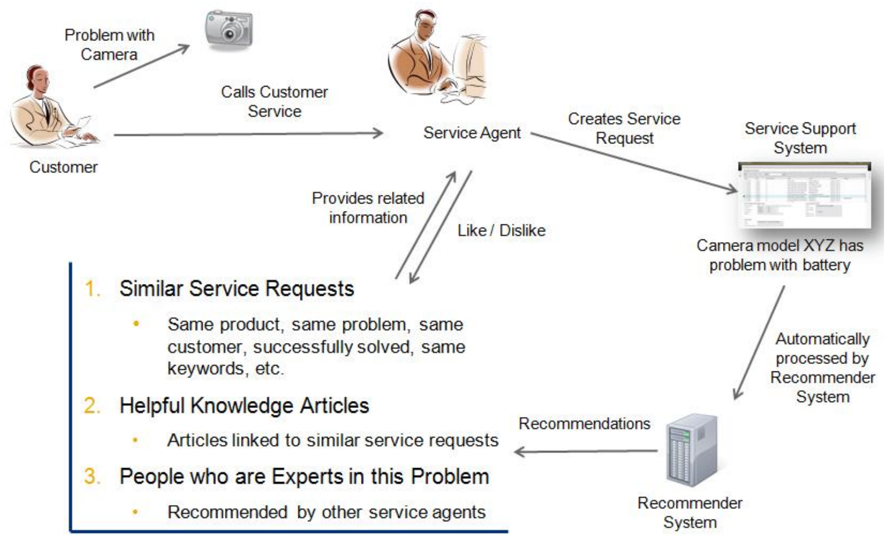
Problem with Camera
Calls Customer Service
Customer
Service Agent
Creates Service Request
Service Support System
Camera model XYZ has problem with battery
Provides related information
Like / Dislike
1. Similar Service Requests
• Same product, same problem, same customer, successfully solved, same keywords, etc.
2. Helpful Knowledge Articles
• Articles linked to similar service requests
3. People who are Experts in this Problem
• Recommended by other service agents
Recommendations
Automatically processed by Recommender System
Recommender System

Fig. 1. User Scenario

3 Demo of the System

A screen recording of the recommender system is available here:

https://vimeo.com/49270529 (password: SAP2012IESD)

4 Addressing Key Themes of the Workshop

4.1 Human Factors and the User Interface

The application provides an end-user interface to domain users who, in this case, are customer support personnel. The UI is designed in a way that the recommendations from the system don't get in the way of the user. The users can choose to leverage the recommendations or ignore them.

Moreover, the UI minimizes the information overload on the user by allowing the user to start from a summarized view of the information and subsequently drill down into more details if needed. The proposed recommendations are categorized in an intuitive way, from similar service requests to related knowledge articles and experts. The UI also provides flexibility to users who want to explore the network of related objects by visually navigating the graph of related objects. Users can search through the business objects that are not discovered by the recommendation engine and then manually add them to the graph. This information becomes available to other users of the system that may work on similar service requests in the future.

In summary, the following human factors have been taken into account:

- The user has control over the process of discovery by evaluating the recommendations and by tagging them with 'like', 'dislike' and 'mark as helpful' features.
- The system provides explanations for why the relationship is proposed and how the weight of the relationship is calculated.
- The system provides a visual interface to let the user navigate the graph freely.

4.2 Computational Models

The application is implemented by overlying a semantic layer on top of an existing object-based layer. With this approach we were able to add semantic capabilities to an existing enterprise application. In the core of the implementation is a home-grown semantic network built as a graph data store and inspired by the Semantic Web technologies (mainly RDF and RDFS).

In order to ensure the scalability of the system, we implemented a hybrid method where part of the graph is persisted in the database and another part of it is always recomputed on-the-fly.

The application employs an algorithm in order to find similar service requests based on a subset of the attributes of the business object representing service requests. The set of attributes used by the similarity algorithm can be configured based on the domain requirements. The criteria for measuring the distance between the attribute values are also customizable. Weighting factors are assigned per each attribute that are taken into account by the algorithm to compute a similarity score between two service requests. A threshold for the similarity score is used by the application to decide which similarity relationships should be persisted in the semantic graph.

An inferencing algorithm calculates new relationships on-the-fly when the user opens up a service request to work on. Domain-specific rules are defined for how new relationships should be inferred. Each rule is defined on a set of business objects. For example, the "RelatedExpert" rule is defined as follows: If Service Request A is similar to Service Request B, and if Service Request B has an Expert C linked to it; then link Expert C to Service Request A as a possibly helpful information.

5 Technical Overview

The core of the system is an implementation of a ‘graph business object’ that captures the semantics of a set of related business objects. Actions can be added to the graph business object for inferring new relationships based on domain-specific rules or heuristics, e.g. knowledge articles linked to the same service request are related. Another set of actions handle the lifecycle of the graph instances such as maintaining the links when new business objects are added or removed.

Each instance of a relationship can have a specific type and weight. The type of relationships can be generic, e.g., calculated, user-defined, etc., or domain-dependent and defined by the domain experts, e.g., related-article., The weight of a relationship is either automatically calculated based on the clustering algorithm or it is set based on rules. One such rule specifies that relationships established by the user have the highest weight. The system learns from the user's input and adjusts the weight of the links to related objects, which in turn changes the order of recommendations provided to the user.

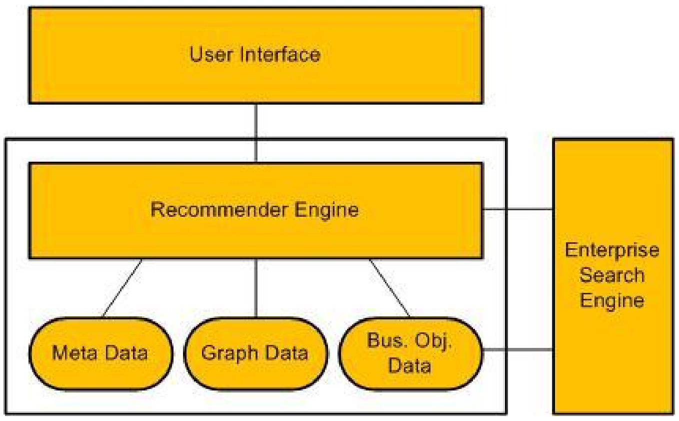

Fig. 2. Overall Architecture

6 Conclusion

Recommender systems are based on discovery of information that might be of help to its users in a specific context. One main challenge in building recommendation systems for enterprise applications is the lack of large volumes of data from which helpful recommendations can be extracted. Relying only on statistical methods for discovering semantic data does not work in most cases. The more effective approach is a hybrid one that combines statistical-based approaches (e.g., clustering) and semantic reasoning approaches (e.g., rule-based inferencing) in order to generate helpful recommendations. The machine-generated recommendations are then fine-tuned by taking into account the human’s feedback.