Proceedings of the International Workshop on Semantic Technologies meet Recommender Systems & Big Data

SeRSy 2012

co-located with the:
11th International Semantic Web Conference (ISWC 2012)

November 11, 2012, Boston, USA

Workshop Organizers and Program Chairs

Marco de Gemmis
Department of Computer Science
University of Bari Aldo Moro, Italy

Tommaso Di Noia
Electrical & Electronics Engineering Department
Politecnico of Bari, Italy

Pasquale Lops
Department of Computer Science
University of Bari Aldo Moro, Italy

Thomas Lukasiewicz
Department of Computer Science
University of Oxford, United Kingdom

Giovanni Semeraro
Department of Computer Science
University of Bari Aldo Moro, Italy

This volume is published and copyrighted by:

Marco de Gemmis
Tommaso Di Noia
Pasquale Lops
Thomas Lukasiewicz
Giovanni Semeraro

This volume appeared online as CEUR-WS.org/Vol-919 at CEUR Workshop Proceedings, ISSN 1613-0073

Preface

These are the proceedings of the *First Workshop on Semantic Technologies meet Recommender Systems & Big Data (SeRSy 2012)*, held in conjunction with the *11th International Semantic Web Conference (ISWC 2012)*.

People generally need more and more advanced tools that go beyond those implementing the canonical search paradigm for seeking relevant information. A new search paradigm is emerging, where the user perspective is completely reversed: *from finding to being found*.

Recommender Systems may help to support this new perspective, because they have the effect of pushing relevant objects, selected from a large space of possible options, to potentially interested users. To achieve this result, recommendation techniques generally rely on data referring to three kinds of objects: users, items and their relations. The widespread success of Semantic Web techniques, creating a Web of interoperable and machine readable data, can be also beneficial for recommender systems. Indeed, more and more semantic data are published following the Linked Data principles, that enable to set up links between objects in different data sources, by connecting information in a single global data space – the Web of Data. Today, the Web of Data includes different types of knowledge represented in a homogeneous form – sedimentary one (encyclopedic, cultural, linguistic, common-sense, …) and real-time one (news, data streams, …). This data might be useful to interlink diverse information about users, items, and their relations and implement reasoning mechanisms that can support and improve the recommendation process.

The challenge is to investigate whether and how this large amount of wide-coverage and linked semantic knowledge can significantly improve the search process in those tasks that cannot be solved merely through a straightforward matching of queries and documents. Such tasks involve finding information from large document collections, categorizing and understanding that information, and producing some product, such as an actionable decision. Examples of such tasks include understanding a health problem in order to make a medical decision, or simply deciding which laptop to buy. Recommender systems support users exactly in those complex tasks.

The primary goal of the workshop is to showcase cutting edge research in the intersection of semantic technologies and recommender systems, by taking the best of the two worlds. This combination may provide the Semantic Web community with important real-world scenarios where its potential can be effectively exploited into systems performing complex tasks.

We wish to thank all authors who submitted papers and all workshop participants for fruitful discussions. We would like to thank the program committee members and external referees for their timely expertise in carefully reviewing the submissions. We would also like to thank our invited speaker Ora Lassila for his interesting and stimulating talk.

October 2012

The Workshop Chairs
Marco de Gemmis
Tommaso Di Noia
Pasquale Lops
Thomas Lukasiewicz
Giovanni Semeraro

Program Committee

Fabian Abel	L3S Research Centre – Germany
Claudio Bartolini	HP Labs @ Palo Alto – USA
Marco Brambilla	Politecnico di Milano – Italy
Andrea Calì	Birkbeck, University of London – UK
Ivan Cantador	Universidad Autónoma de Madrid – Spain
Pablo Castells	Universidad Autónoma de Madrid – Spain
Federica Cena	University of Turin – Italy
Bettina Fazzinga	Università della Calabria – Italy
Tim Furche	University of Oxford – UK
Nicola Henze	Leibniz Universität Hannover – Germany
Dominikus Heckmann	DFKI – Germany
Leo Iaquinta	Univ. di Milano Bicocca – Italy
Roberto Mirizzi	HP Labs @ Palo Alto – USA
Ahsan Morshed	CSIRO – Australia
Enrico Motta	Open University in Milton Keynes – UK
Cataldo Musto	Università di Bari "Aldo Moro" – Italy
Fedelucio Narducci	Univ. di Milano Bicocca – Italy
Vito Claudio Ostuni	Politecnico of Bari – Italy
Alexandre Passant	seevl.net – Ireland
Gerardo I. Simari	University of Oxford – UK
Markus Zanker	Alpen–Adria–Universität Klagenfurt – Austria

Table of Contents

Link Prediction in Multi-relational Graphs using Additive Models. 1-12
Xueyan Jiang, Volker Tresp, Yi Huang and Maximilian Nickel

Driver Recommendations of POIs using a Semantic Content-based Approach. 13-24
Rahul Parundekar and Kentaro Oguchi

Semantic Network-driven News Recommender Systems: a Celebrity Gossip Use Case. 25-36
Marco Fossati, Claudio Giuliano and Giovanni Tummarello

Cinemappy: a Context-aware Mobile App for Movie Recommendations boosted by DBpedia. 37-48
Vito Claudio Ostuni, Tommaso Di Noia, Roberto Mirizzi, Romito Davide and Eugenio Di Sciascio

Ontology-based Rules for Recommender Systems. 49-60
Jeremy Debattista, Simon Scerri, Ismael Rivera and Siegfried Handschuh

Ontology-centric Decision Support. 61-72
Marco Rospocher and Luciano Serafini

RING: A Context Ontology for Communication Channel Rule-based Recommender System. 73-82
Miguel Lagares Lemos, Daniel Villanueva Vasquez, Mateusz Radzimski, Angel Lagares Lemos and Juan Miguel Gómez-Berbís

Link Prediction in Multi-relational Graphs using Additive Models

Xueyan Jiang[2], Volker Tresp[1,2], Yi Huang[1,2], and Maximilian Nickel[2]

[1] Siemens AG, Corporate Technology, Munich, Germany
[2] Ludwig Maximilian University of Munich, Munich, Germany

Abstract. We present a general and novel framework for predicting links in multirelational graphs using a set of matrices describing the various instantiated relations in the knowledge base. We construct matrices that add information further remote in the knowledge graph by join operations and we describe how unstructured information can be integrated in the model. We show that efficient learning can be achieved using an alternating least squares approach exploiting sparse matrix algebra and low-rank approximations. We discuss the relevance of modeling nonlinear interactions and add corresponding model components. We also discuss a kernel solution which is of interest when it is easy to define sensible kernels. We discuss the relevance of feature selection for the interaction terms and apply a random search strategy to tune the hyperparameters in the model. We validate our approach using data sets from the Linked Open Data (LOD) cloud and from other sources.

1 Introduction

There is a growing amount of data published in multirelational graphs where information elements are represented as subject-predicate-object (s, p, o) triples. Entities (i.e., subjects and objects) are represented as nodes and statements are represented as directed labeled links from subject node to object node. A machine learning task of some generality is the prediction of links between entities using patterns in known labeled links in the knowledge base.

We present a general framework for predicting links in multirelational graphs using a set of matrices describing the various instantiated relations in the knowledge base. We first consider triples in the immediate neighborhood of the triple of interest and then construct matrices that add information further remote in the knowledge graph by performing join operations. We also consider the case that unstructured information is available that can support the link prediction task and we describe how unstructured information can be integrated in the model. Examples of unstructured information are textual documents describing the involved entities (e.g., from the entities' Wikipedia pages). We show that efficient learning can be achieved using an alternating least squares approach exploiting sparse matrix algebra and low-rank approximations. We discuss the relevance of modeling nonlinear interactions and add corresponding model components. We also discuss a kernel solution which is of interest when it is easy to define

sensible kernels. We discuss the relevance of feature selection for the interaction terms and apply a random search strategy to tune the hyperparameters in the model. We validate our approach using data sets from the Linked Open Data (LOD) cloud and from other sources.

The paper is organized as follows. The next section discusses related work. Section 3 describes our basic approach. Section 4 describes the cost function and the alternating least squares solution for parameter learning. In Section 5 we discuss aggregation via joint operations, the inclusion of unstructured information and interaction terms. Section 6 contains our experimental results. Section 7 presents our conclusions.

2 Related Work

One of the first line of research where matrix representations were used for link prediction in multirelational graphs is the SUNS framework [10] [5]. A major extension was the probabilistic extension of the SUNS approach reported in [7]. The same paper also describes how information extraction (IE) can be combined with deductive reasoning and machine learning for link prediction, where the combination is implemented as a postprocessing step. The additive approach presented in this paper is novel and has several advantages. It considers a model for a complete knowledge-base of triples and considers dependencies on all triples in the immediate neighborhood of the triple. Whereas in [7], the combination was a postprocessing step, here we optimize the additive model globally. Also the discussion on aggregation by joint operations is novel, as well as the application of alternating least squares for optimizing the penalized cost function.

The winning entries in the Netflix competitions are based on matrix factorization [1]. The main difference is that in those applications unknown ratings can be treated as missing entries whereas in relational prediction, the topic here, they are treated as negative evidence.

Multi-relational graphs also map elegantly to a tensor representation. Tensor models for relational learning have been explored in [9], showing both scalability and state-of-the-art results on benchmark datasets.

3 Link Prediction in Multi-relational Graphs

3.1 Relational Adjacency Matrices

In this paper we assume that labeled links are represented as triples of the form (s, p, o) where subject s and object o stand for entities in a domain and where p is the predicate, i.e. the link label. We define a variable $x_{i,j,k}$ that is associated with the triple (s $= i$, p $= j$, o $= k$). We set $x_{i,j,k} = 1$ when the triple is known to exist, otherwise $x_{i,j,k} = 0$. In the multirelational graph, the entities form the nodes and the existing triples form labeled links.

We now consider a domain with N entities and P predicates. For the predicate p$= j$ in the domain we define a relational adjacency matrix $X_j \in \mathbb{R}^{N \times N}$

where $(X_j)_{i,k} = 1$ if $x_{i,j,k} = 1$ and $(X_j)_{i,k} = 0$ otherwise. The matrix of concatenated relational adjacency matrices $X = (X_1, \ldots X_P)$ describes all existing and all potential triples involving all known entities in the knowledge base.

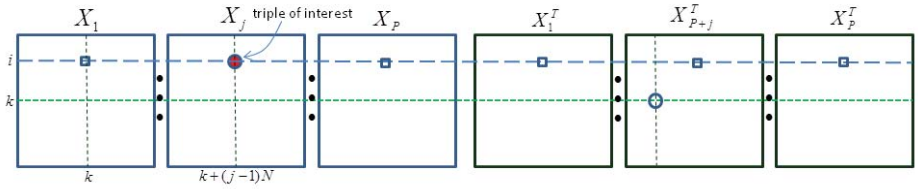


Fig. 1. The figure shows the matrix M and illustrates the terms in Equation 1. We assume that the goal is to predict the matrix entry $\hat{x}_{i,j,k}$ in matrix X_j indicated by the small red circle. The terms in the first sum (subject term) are represented by the upper dashed blue line, the terms in the second sum (object term) are represented by the lower green dotted line, and the terms in the third sum (subject-object term) correspond to the small rectangles.

3.2 The Basic Model

In the basic model we assume that the truth value of a triple (s= i, p= j, o= k) can be estimated as a linear combination of directly related triples, defined as all triples (s= i, p= j', o= k') where i is the subject, all triples (s= i', p= j', o= i) where i is the object, all triples (s= k, p= j', o= k') where k is the subject and all triples (s= i', p= j', o= k) where k is the object. Finally we consider triples with arbitrary predicates but where two entities from the target triple are involved, i.e., (s= i, p= j', o= k), and (s= k, p= j', o= i). If $x_{i,j,k} = 0$, the predicted $\hat{x}_{i,j,k} \geq 0$ can then be interpreted as a likelihood that the triple is true based on the immediate context of the triple.

We now form the matrix $M = (X, X^\dagger)$ where $X^\dagger = (X_1^T, \ldots X_P^T)$ denotes the in-place matrix transposed of X (Figure 1). Let $(M)_{i,l} = m_{i,l}$.

Following the discussion we form the model

$$\hat{x}_{i,j,k} = \sum_{l=1}^{2PN} w_{l,k+(j-1)N}\, m_{i,l} + \sum_{l=1}^{2PN} r_{l,i+(j-1)N}\, m_{k,l} + \sum_{l=1}^{2P} h_{l,j}\, m_{i,k+N(l-1)} \quad (1)$$

For further reference, we call the first term in the sum in Equation 1 the subject term, the second one the object term, and the last one the subject-object term.[3] The $w_{,,}$, $r_{,,}$, and $h_{,,}$ are model parameters to be estimated.

[3] Note that we get nontrivial solutions by using regularized parameter fits with low-rank constraints, as described in Section 4.

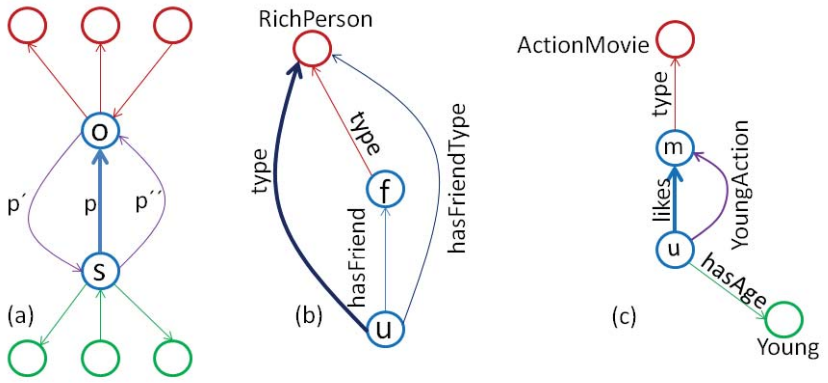


Fig. 2. (a): The goal is to predict the likelihood of the triple (s, p, o). In Equation 1, the triples attached to s correspond to the subject term, the triples attached to o correspond to the object term, and the triples linking s and o correspond the subject-object term. (b): From the triple (u, hasFriend, f) and (f, type, richPerson) we derive via aggregation (s, hasFriendType, RichPerson) which can be useful to predict (u, type, richPerson). (c): From (u, hasAge, Young) and (m, type, ActionMovie) we derive (u, youngAction, m) which is useful for predicting (u, likes, m).

The subject term represents triples where i is the subject, when $l = 1, \dots P$ or where i is the object, when $l = P+1, \dots 2P$. Similar the object term represents triples where k is the subject, when $l = 1, \dots P$ or where k is the object, when $l = P+1, \dots 2P$. The subject-object term considers all triples that involve both i and k with any predicate (see Figure 2 (a)).

3.3 Model Discussion

Note that we assume that the rows in M are exchangeable such that the weights in object term $w_{l,k+(j-1)N}$ are independent of i, and the weights in the subject term $r_{l,i+(j-1)N}$ are independent of k. The parameters $h_{l,j}$ in the subject-object term are independent of both i and k.

There are of course also other ways to segment the parameter space. For example, one might decide that the semantics of the predicate "like" is very different when subject is a person than if the subject is a dog and the object is a bone. Technically this could mean that, e.g., we write $w_{l,k+(j-1)N,type(i)}$ and the model correspondingly would have more parameters.

As a special case, we only have one predicate, i.e. "like" and entity types users and movies. If we apply the learning procedure as described in Section 4 we obtain a solution that only exploit correlations between triples with the same predicate, i.e., intrarelational correlations. In effect, we obtain a regularized low-rank approximation of the relational adjacency matrix which is a model often

used in collaborative filtering applications. Additional relational adjacency matrixes, for example representing user and movie attributes, can then help to support the prediction of "like"-triples.

One might want to think of the last equation in terms of an if-then-rule where the right side of the equation describes the condition and the left side describes the conclusion. In this view the subject ?s and the object ?o would be variables[4] and the subject term describes relations including the first variable, the object term describes relations including the second variable, and the subject-object term describes relations including both variables. All these variables are universally quantified, which means that the expression is valid for all subjects ?s and all objects ?o. We can introduce additional variables in the condition part for certain aggregation operations, as described in Section 5, and these variables would be existentially quantified (as in Horn clauses).

4 Cost Function and Parameter Optimization

4.1 Penalized Cost Function

We can write the model of Equation 1 efficiently in matrix form as

$$\hat{X} = MW + (MR)^{\dagger} + \text{matrix}_{(N \times PN)} \left(\tilde{M} \; H \right) \tag{2}$$

Here, $\tilde{M} = (\text{vect}(X_1), \ldots, \text{vect}(X_P), \text{vect}(X_1^T), \ldots, \text{vect}(X_P^T))$ is an $N^2 \times 2P$ matrix where the column vector vect(.) contains all elements of the corresponding relational adjacency matrix. Furthermore, $W \in \mathbb{R}^{2PN \times PN}$, $R \in \mathbb{R}^{2PN \times PN}$, and $H \in \mathbb{R}^{2P \times P}$ are parameter matrices. The operation $\text{matrix}_{(N \times PN)}(.)$ transforms the result of the matrix product into a $N \times PN$ matrix.

We define a penalized least squares cost function as

$$\|X - \hat{X}\|_F^2 + \lambda_W \|W\|_F^2 + \lambda_R \|R\|_F^2 + \lambda_H \|H\|_F^2$$

where $\|\|_F$ is the Frobenius norm. The last three terms are used to regularize the solution to avoid overfitting.

4.2 Alternating Least Squares

We optimize the parameter matrices W, R, and H using an alternating least squares procedure as described in this subsection.

To reduce computation and to further regularize the solution, we first decompose using singular value decomposition (SVD)

$$M = UDV^T \qquad \tilde{M} = \tilde{U}\tilde{D}\tilde{V}^T$$

and only use the leading singular values and corresponding singular vectors in the model. Another benefit of this low-rank approximation is that we implicitly

[4] We use the common notation of indicating a variable by a question mark in front of a symbol.

benefit from a sharing of statistical strengths leading to performance improvements, as it is well known from Latent Semantic Analysis (LSA).

We define $\hat{X}^{(-W)}$ as X minus the estimate in Equation 2 using the parameter estimates in the current iteration step, except that $W = 0$, i.e. we remove the subject term in the sum. Similarly, we define $\hat{X}^{(-R)}$ as X minus the estimate in Equation 2 using the parameter estimates in the current iteration step, except that $R = 0$, i.e. we remove the object term in the sum. Finally, we define $\hat{X}^{(-H)}$ as X minus the estimate in Equation 2 using the parameter estimates in the current iteration step, except that $H = 0$, i.e. we remove the subject-object term in the sum.

In the alternating least squares steps we iterate until convergence

$$MW = U_r \operatorname{diag}\left\{\frac{d_i^2}{d_i^2 + \lambda_W}\right\}_{i=1}^{r} U_r^T \hat{X}^{(-W)} \tag{3}$$

$$MR = U_r \operatorname{diag}\left\{\frac{d_i^2}{d_i^2 + \lambda_R}\right\}_{i=1}^{r} U_r^T \left(\hat{X}^{(-R)}\right)^{\dagger} \tag{4}$$

$$\tilde{M}H = \tilde{U}_{\tilde{r}} \operatorname{diag}\left\{\frac{\tilde{d}_i^2}{\tilde{d}_i^2 + \tilde{\lambda}_H}\right\}_{i=1}^{\tilde{r}} \tilde{U}_{\tilde{r}}^T \tilde{X}^{(-H)} \tag{5}$$

where in $\tilde{X}^{(-H)} = \text{matrix}_{N^2 \times P} X^{(-H)}$ each relational adjacency matrix is written as a column vector. In particular for the update in Equation 5, a solution in terms of the V-matrices might be more efficient (see the Appendix).

4.3 Computational Costs

Considering domains with several million entities, the computations seem to be expensive. Fortunately, in the computations one can explore the extreme sparsity of all relational adjacency matrices in many domains of interests. For example, due to type constraints, nonzero elements are often restricted to one or a small number of blocks in the matrices. For example if entities are users and movies and X stands for "likes" then only the submatrix with users as rows and movies as columns contains nonzero elements, reflecting the fact that users like movies but, e.g., movies do not like users. Also, if one is only interested to predict entries in one particular relational adjacency matrix, we only need to calculate the parameters relevant to predicting the entries in that particular matrix.

We also want to point out that one could also apply the SVD to each relational adjacency matrix separately or to blocks of relational adjacency matrices, instead of M; essentially one should make this decision on the expected performance benefits of the matrix decompositions and the computational costs.

5 Extensions

5.1 Aggregation by Join Operations

The triples represented in Equation 2 only consider the immediate neighborhood of the triples (s, p, o) under consideration. It is easy to extend the formalism

to also consider triples further away in the graph. As an example, consider the case that the likelihood that a person likes a movie is increased if at least one friend likes the movie. The latter information can be represented by the matrix, representing a join operation, formed by $X_{\text{friendLikesMovie}} = \min(1, X_{\text{friendOf}} X_{\text{likes}})$ where min is applied component wise. Then $X_{\text{friendLikesMovie}}$ (and its transposed) is simply added as an additional relational adjacency matrix. Now we can model (via the subject term in Equation 2) that a person might like "Action Hero 3" if at least one friend likes "Action Hero 3"; the subject-object term in Equation 2 can even model the more general dependency that a person likes any movie if at least one friend likes that movie.

The general form of an aggregated adjacency matrix is $X_a = \min(1, \prod_i M_i)$ where $M_i \in (X_1, \dots X_P, X_1^T, \dots X_P^T)$. Naturally, it is not feasible to consider an infinite set of matrix products. Possible approaches are that the user defines a small set of interesting candidates or that one applies structural search, e.g., by using approaches borrowed from the field of Inductive Logic Programming. Many more forms of aggregation are possible. For example one might not apply the min operation and, e.g, count how many friends liked a movie, or what percentage of friends liked a movie.

Here are two interesting examples involving join operations. First, let's assume that a person tends to be rich if this person has a rich friend: The triple of interest is (?u, type, RichPerson). We join (?u, hasFriend, ?f) and (?f, type, RichPerson) and obtain a matrix that indicates if anybody of a person's friends is rich (see Figure 2 (b)). Second, let's assume that a person often prefers restaurants of the nationality of that person: The triple of interest is (?u, likes, ?r). We join (?u, hasNationality, ?c) and (?r, hasNationality, ?c) and obtain a matrix that indicates if the user and the restaurant have the same nationality.

5.2 Contextual and Unstructured Data

Sometimes there is contextual information available, often in textual form, that describe entities and relationships and can be exploited for link prediction [7]. For example, one can use keywords in an entity's Wikipedia articles as attributes of that entity. The triples (s, itsWikiPageHasKeyword, Keyword) can simply be added as an additional relational adjacency matrix in the approach. If a keyword can be identified as an entity, then this information is even more valuable. Information extraction (IE) can also be used to extract triples from text and these triples can then presented in matrix form as well. In the latter case, the subject-object term in Equation 2 can be expected to be most valuable: if, for example, the IE system extracts with high confidence that (Jack, knows, Jane) this could be information for predicting that (Jack, hasFriend, Jane).

5.3 Interaction Terms

In Equation 2 we used a linear system, which is suitable in many high-dimensional domains. Of course, one can apply more general models such as neural networks as predictive models. Often this is unsuitable since the computational costs would

explode. In our approach we stay with a model linear in the parameters but add nonlinear interaction terms. As an example, assume that young users prefer action movies. We define a new triple (?u, YoungAction, ?m) that is true if (?u, hasAge, Young) is true and (?m, type, ActionMovie) is true (see Figure 2 (c)). In general, the subject-object term in Equation 1 can be expected to be most valuable here as well. To keep the number of these interaction terms small, we apply a feature selection procedure, as described in Section 6.

5.4 Kernel Formulation

So far our discussion focussed on a representation in feature space. Here we discuss a representation in kernel space. A kernel formulation is appropriate for data in a multirelational graph and suitable kernels are described in [11,4,3,8].

From Equation 2 one can see that two kernels are involved in our approach, the first one $k(.,.)$ involving two entities, either two subjects $k(s,s')$ or two objects $k(o,o')$. The second kernel $\tilde{k}((s,o),(s',o'))$ involves two subject-object pairs. Given the corresponding kernel matrices K and $\tilde{K}$ we can decompose using a singular value decomposition

$$K = UDD^TU^T \quad \tilde{K} = \tilde{U}\tilde{D}\tilde{D}^T\tilde{U}^T$$

and use the resulting terms in the update Equations 3 to 5.

6 Experiments

6.1 Tuning of Hyperparameters

We have several hyperparameters that need to be tuned (rank of approximations; regularization parameters). We follow the approach described in [2] and perform a random search for the best hyperparameters using cross-validation sets (i.e. they are not tuned on the test set).

6.2 Synthetic Data

The synthetic data has been generated according to our modeling assumptions. We define a target predicate of interest and call the triples involving the target predicate the target triples. In addition we have triples related to the subject, i.e., describing subject attributes, and triples related to the object, i.e., describing object attributes. In addition we use interaction triples generated by conjunctions on subject and object triples.

Figure 3 shows the results of using different relational adjacency matrices. The proposed model that uses all sources of information (M_{global}) performs best. Also if we only exploit subject attributes and object attributes (F_{all}) we obtain significant predictive power. A model only using intrarelational information M_{CF} is quite strong. The reason is that, if sufficient amount of target triples are known to be true, the information on subject and object attributes is implicitly modeled

in M_{CF} as well. This is a result also confirmed in the remaining experiments: if M_{CF} is quite strong, adding subject and object information does not improve the model further, even when the latter might have predictive power. The proposed model (M_{global}) is significantly better than the reference model (M_{HBS}) described in [7] that used a hierarchical Bayesian combination scheme.

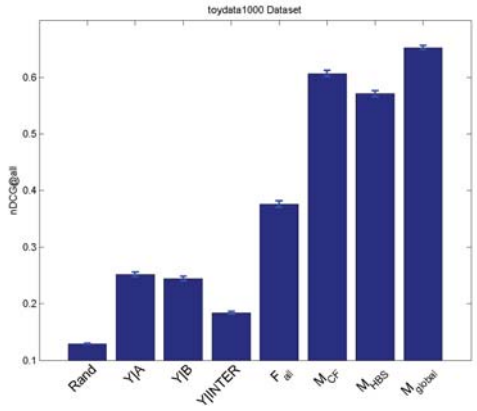


Fig. 3. Test results on synthetic data. For each subject entity in the data set, we randomly selected one true relation to be treated as unknown (test statement). In the test phase we then predicted all unknown relations for the entity, including the entry for the test statement. The test statement should obtain a high likelihood value, if compared to the other unknown entries. The normalized discounted cumulative gain (nDCG@all) [6] is a measure to evaluate a predicted ranking. We see that the proposed method (M_{global}) is significantly better than the model that only relies on intrarelational correlations (M_{CF}). The reference model (M_{HBS}) does not significantly improve w.r.t. M_{CF}. Predictions only based on subject attributes $F_{Y|A}$ only based on object attributes $F_{Y|B}$, and only based on interaction terms $F_{Y|INTER}$ are much better than random. F_{all} uses subject attributes, object attributes and interaction terms, but not intrarelational correlations in the target triples.

6.3 Associating Diseases with Genes

The task here is to predict diseases that are likely associated with a gene based on knowledge about gene and disease attributes and about known gene-disease patterns. In our experiments we extracted information on known relationships between genes and diseases from the LOD cloud, in particular from Linked Life Data and Bio2RDF, forming the triples (Gene, related_to, Disease). In total, we considered 2462 genes and 331 diseases. We retrieved textual information describing genes and diseases from corresponding text fields in Linked Life Data and Bio2RDF.

We have 49801621 potential interaction terms which we reduced to 1132 by using a fast feature selection procedure evaluating the Pearson correlation between targets and interaction term.

Figure 4 (left) shows the results for predicting diseases for genes. Our proposed model gives very good results, although the reference model is slightly stronger. Figure 4 (right) shows the results for predicting genes for diseases. Due to sparsity, this task is more difficult and our proposed model performs best.

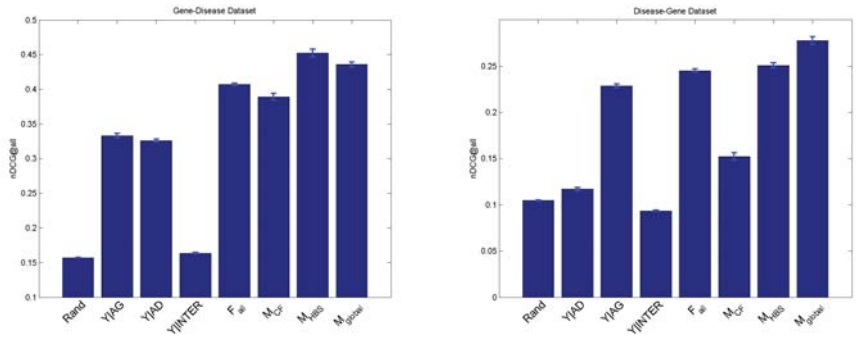


Fig. 4. The goal is to predict the relationship between genes and diseases. On the left we ranked recommended diseases for genes and on the right we ranked genes for diseases. In the left experiment, the subject attributes of the genes $F_{Y|AG}$, and of the object attributes of the diseases $F_{Y|AD}$ are comparable in strength. F_{all} that uses gene attributes, disease attributes and interaction terms in combination gives strong results. Our proposed model (M_{global}) can exploit both contextual information and intrarelational correlations. The reference model (M_{HBS}) is slightly stronger than our proposed model. The right plot shows results from the second experiment where we rank genes for diseases. This task is more difficult due to the large number of genes and our proposed system gives best results.

6.4 Modeling MovieLens Data

We used 943 users and 1600 movies from the MovieLens data set[5] and evaluated if a user has seen a movie or not. 99 user attributes were derived from age (5 classes), gender (2 classes), occupation (21 classes), and the first two digits of the ZIP code. The 89 movie attributes were derived from genre, release month and release year. Figure 5 (left) shows the results. Although the attribute information on the movies and the users have predictive power (significantly above random), a model exploiting intrarelational correlations (M_{CF}) gives very good performance and the proposed model and the reference model cannot improve beyond the performance of M_{CF}. As in the experiment on the synthetic data,

[5] http://www.grouplens.org/node/73

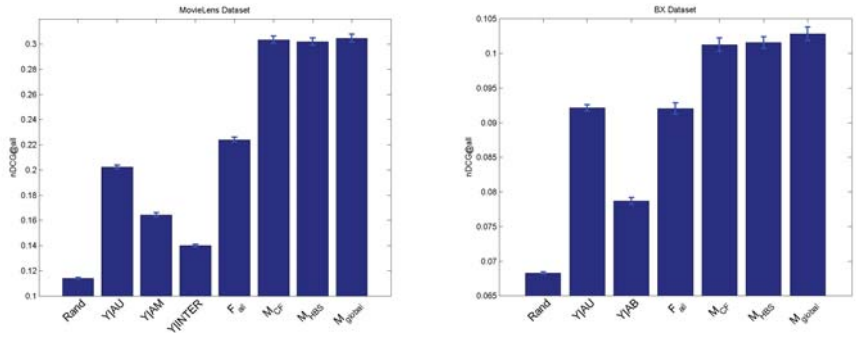


Fig. 5. Left: Experiments on movielens data. $F_{Y|AM}$ describes the modeling performance using only the attribute information on the movies and $F_{Y|AU}$ describes the modeling performance using only the attribute information on the users. Although the attribute information on the movies and the users have predictive power (significantly above random), a model exploiting intrarelational correlations (M_{CF}) gives very good performance and the proposed model and the reference model cannot significantly improve beyond the performance of M_{CF}. As in the experiment on the synthetic data, there is information in the attribute data but this information is also represented in M_{CF}. Right: Experiments on the book-crossing data set. We see the same trends as in the movielens experiments although the proposed approach seems to improve on the model M_{CF}, which only exploits intrarelational correlations.

there is information in the contextual data but this information is also represented in M_{CF}. We have 8811 potential interaction terms which we reduced to 200 by using a fast feature selection procedure evaluating the Pearson correlation between targets and interaction term.

6.5 Modeling Book Preferences

We used the BookCrossing data set[6] to predict if a user rated a book. The data set consisted of 105283 users and 340554 books. A user is described by 5849 attributes (derived from age and city, province and country) and a book is described by 24508 attributes (authors, publication year, publisher). The goal is to predict if a user would rate (i.e., read) a book. The results in Figure 5 (right) show that the proposed modeling approach gives best results.

7 Conclusions

We have presented a general framework for predicting links in multirelational graphs. We showed that efficient learning can be achieved using an alternating least squares approach.

[6] http://www.bookcrossing.com

The approach can be extended in several directions. First, for the entries in the relational adjacency matrices one can use real numbers, e.g., between zero and one, and the user can represent the certainty that a triple is true [7]. Second, we can exploit deductive reasoning by calculating the deductive closure prior to learning [7]. Third, the prediction in Equation 1 can be applied recursively permitting global information flow through the relational graph. Finally, we can easily generalize to entities not in the training set, either by using Equations 3 to 5 directly or by transforming these equations into appropriate equivalent forms.

References

1. Robert M. Bell, Yehuda Koren, and Chris Volinsky. All together now: A perspective on the netflix prize. *Chance*, 2010.
2. James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 2012.
3. Stephan Bloehdorn and York Sure. Kernel methods for mining instance data in ontologies. *ESWC*, 2007.
4. Thomas Gärtner, John W. Lloyd, and Peter A. Flach. Kernels and distances for structured data. *Machine Learning*, 2004.
5. Yi Huang, Markus Bundschus, Volker Tresp, Achim Rettinger, and Hans-Peter Kriegel. Multivariate structured prediction for learning on the semantic web. In *ILP*, 2010.
6. Kalervo Järvelin and Jaana Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *SIGIR'00*, 2000.
7. Xueyan Jiang, Yi Huang, Maximilian Nickel, and Volker Tresp. Combining information extraction, deductive reasoning and machine learning for relation prediction. In *ESWC*, 2012.
8. Ute Lösch, Stephan Bloehdorn, and Achim Rettinger. Graph kernels for RDF data. *ESWC*, 2012.
9. Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011.
10. Volker Tresp, Yi Huang, Markus Bundschus, and Achim Rettinger. Materializing and querying learned knowledge. In *IRMLeS*, 2009.
11. S. V. N. Vishwanathan, Nic Schraudolph, Risi Imre Kondor, and Karsten Borgwardt. Graph kernels. *Journal of Machine Learning Research - JMLR*, 2008.

8 Appendix

We only derive Equation 3. The derivations for Equations 4 and 5 are equivalent. We start with the regularizes least squares solution for estimating $\hat{X}^{(-W)}$ based on M

$$MW = M(M^T M + \lambda_W I)^{-1} M^T \hat{X}^{(-W)}$$

If we use the low-rank approximation $M \approx U_r D_r V_r^T$, where $D_r = \text{diag}\,\{d_i\}_{i=1}^r$, we get

$$MW = U_r D_r V_r^T (V_r D_r U_r^T U_r D_r V_r^T + \lambda_W I)^{-1} V_r D_r U_r^T \hat{X}^{(-W)}$$

$$= U_r \text{ diag} \left\{ \frac{d_i^2}{d_i^2 + \lambda_W} \right\}_{i=1}^r U_r^T \hat{X}^{(-W)} = \hat{X}^{(-W)} V_r \text{ diag} \left\{ \frac{d_i^2}{d_i^2 + \lambda_W} \right\}_{i=1}^r V_r^T$$

Driver Recommendations of POIs using a Semantic Content-based Approach

Rahul Parundekar and Kentaro Oguchi

Toyota InfoTechnology Center, U.S.A.
Mountain View, CA
{rparundekar,koguchi}@us.toyota-itc.com

Abstract. In this paper, we present a semantic content-based approach that is employed to study driver preferences for Points of Interest (POIs), e.g. banks, grocery stores, etc., and provide recommendations for new POIs. Initially, logs about the places that the driver visits are collected from the cloud-connected navigation application running in the car. Data about the visited places is gathered from multiple sources and represented semantically in RDF by 'lifting' it. This semantic data is then combined with driver context and then input into a machine learning algorithm that produces a probabilistic model of the driver's preferences of POIs. When the driver searches for POIs in an unknown area, this preference model is used to recommend places that he is most likely to prefer using a nearest-neighbor approach. In this paper, we describe the details of this content-based approach for recommendation, along with the results of a user study that was conducted to evaluate the approach.

Keywords: Semantic Web, Recommendations, Preference Modeling

1 Introduction

In deploying connected services to the car, it is important that driver safety is given a high priority. To reduce possible distractions in accessing information, the in-vehicle navigation system enforces restrictions on the way information is presented to the driver. One such constraint is that the number of items that can be displayed in a list on a single screen is fixed to a limited number of slots. When the driver searches for banks in the car, for example, the search results are displayed as a list filled in the available slots. If the number of search results exceeds the number of slots, then the extra results are pushed to the next page. Accordingly, there arises a need for presenting the most relevant information to the driver in those slots. We model this, as a recommendation problem of providing personalized, contextualized information to the driver. In particular, this paper discusses the recommendation of Points of Interest (POIs), e.g. banks, grocery stores, etc., in the car using a combination of semantic technologies and a recommendation system.

In our previous paper on Learning Driver Preferences of POIs using a Semantic Web Knowledge System[8], we presented the architecture and components of

a semantic system that is able to model the driver's preferences. In this paper, we present details of the recommendation aspect of the work including the reasons for the selection of a content-based approach, the details of the nearest-neighbor recommendation method and the results of a user study that was conducted earlier this year. Using the history of Points of Interests (POIs) that the driver visits, we can build a model of the places he/she (henceforth referred to as *he*) is more likely to prefer. For example, the driver may have certain preferences for banks. Based on the ones he has visited in the past, we try to build a preference model that can be used to determine his affinity for a bank he has not visited before. The next time he is searching for a bank in an unfamiliar place, his preference model is used to recommend him one from the banks around him.

The paper is organized as follows. We first provide a detailed explanation of the recommendation task and describe the reasons for the selection of a content-based approach. We then describe how we learn the driver preferences. This is followed by an explanation of how the learned model is used in the recommendation of POIs. We also briefly describe the underlying system, which uses semantic technologies in the data representation and services, called the Semantic User Preference Engine or *Supe*. This is followed by a description of the user study that was conducted using an implementation of Supe, and its results. We also include relevant work in the Semantic Web and recommendation literature, for modeling user preferences and recommendations. Lastly, we conclude by summarizing our findings along with future work.

2 Using a Content-based Approach for Recommendation

The selection of the algorithm for POI recommendation in the car is largely based on two issues: the nature of the data available and desired recommendation to be produced. In the first case, data comes from the usage history of the navigation application in the vehicle. The user can select a place to navigate to in three ways, as can be seen in Fig. 1: (1.a) driver chooses a POI from the head-unit; (1.b) user selects a POI from a suite of installed applications on his smartphone or (1.c) user pre-selects a POI from his desktop/browser application and 'sends' the POI over the Cloud to the navigation application running in the car. The selected POIs from these connected devices can be tracked on the Server running in the Cloud before sending it to the navigation application. As we can only track places that the driver has visited, which we assume he likes, we only have positive training examples. For the second issue, we want to try to provide a recommendation that answers the question, 'Which POIs among the ones available around the driver is he most likely to prefer?', when the user searches for POIs. Accordingly, the recommender system should be able to choose the place most likely to be preferred by the user from a set of candidate places returned by the database.

Recommender systems are popularly classified into *Collaborative Filtering*, *Content-based* approaches or a mixture thereof[2, 1]. Common issues with the *Collaborative Filtering* algorithms are the new user problem, new item problem and the sparsity problem. Out of the three, data sparsity is the biggest possible

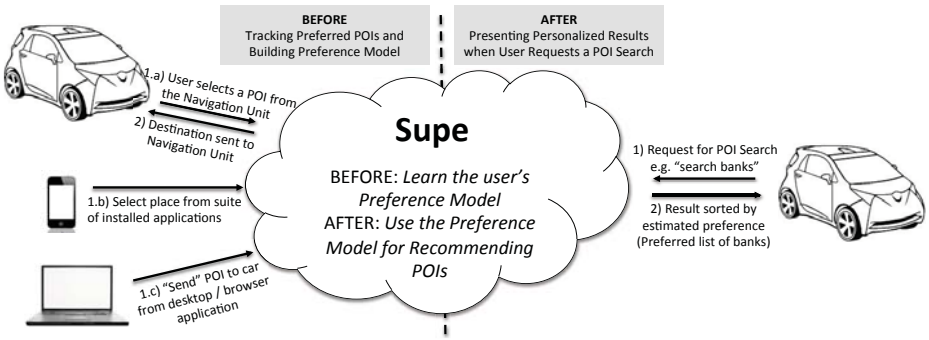


Fig. 1. Recommendation Task Overview

reason of failure of recommendation in the dataset. As an example, consider a person who lives in San Jose, CA. The places that he visits frequently in his neighborhood - e.g. his bank, grocery store, etc., along with history of others in his neighborhood become part of the training dataset. When the user travels to a remote place - e.g. Livermore, CA (about 40 miles away) and wants to search for a bank, it is highly unlikely that there would be another set of users who go to the same bank as the user in San Jose *and* might have visited a bank in that particular neighborhood in Livermore. Contextual information is also important in providing relevant recommendations. Though *Collaborative Filtering* techniques that include context by introducing new dimensions apart from the traditional two - *Users* and *Items* - have been studied[2], they suffer from worse data sparsity problems despite optimizations for computational overhead.

In contrast, a *content-based* approach seems more intuitive. In the first example, the user might prefer a certain banking chain, which has a branch in his neighborhood and he would prefer a local branch when he is searching for ATMs in Oakland. In case of gas stations, he may have a preference for cheaper gasoline. A single *Collaborative Filtering* approach would be insufficient for recommending such cases, especially in the case of fluctuating gas prices. Contextual information can be appended to the POI data as extended attributes to generate a context-specific version of the item, which can then be used to provide recommendations. The content-based approach for POI recommendation employed in this paper, is similar to other recommender systems in relevant literature [11, 6] and is described in the following sections.

3 Generating the Preference Model From Driver History

Before we can recommend POIs to the driver, we first need to generate a model of his preferences. To do so, the data collected from his navigation history is first converted into machine learnable data. When connected devices send data to the in-vehicle navigation application or the driver selects a destination on the navigation system, Supe tracks the visited/consumed POIs and stores them as driver history. The process of generation of the user's preference model from this

data is described below. Fig. 2 shows the steps involved in converting POI data about a bank, visited by the driver, into machine learnable data.

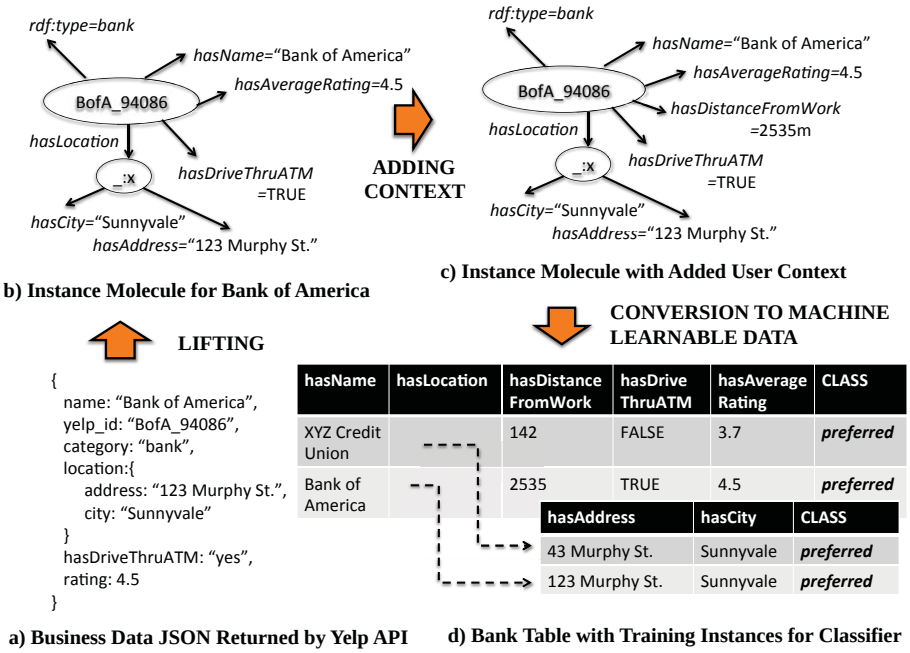


Fig. 2. Building User Preferences (Note: Only a subset of the actual attributes used is shown.)

Fetching RDF data for the Visited POIs: Data for the POIs, which were collected as driver history, is retrieved from multiple sources like a POI database or Web Services (by searching for the POI using information like its name, location, address, etc.). For example, the JSON response from a Web Service (e.g. Yelp API) for the "Bank of America" branch, which the user visited, is shown in Fig. 2 (a). As each of these sources might have different schemas, it is necessary to integrate this information together into a common vocabulary. At the heart of Supe is a Places Ontology (see Fig. 3). The ontology defines a concept hierarchy of places along with the properties associated with each concept. Data from the different sources is 'lifted' into this ontology and merged together. For the "Bank of America" that the user visited, the lifting process converts the JSON response of the Web Service into an RDF instance molecule [4] as shown in Fig. 2 (b). Because of the simplicity of the POI domain, the lifting rules were determined at design time.

Adding Context: Data from the driving history along with the driver's personal information is then used to automatically generate context information. The relevant triples, generated using pre-defined rules, are then added to the RDF instance molecule from the previous step. For example, if the user has his home or work address stored, then the *distance from home* or *distance from work*

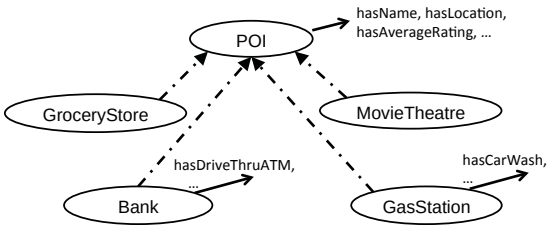


Fig. 3. Place Ontology (partial)

contexts can be added. User context *hasDistanceFromWork* is added to the instance molecule in Fig. 2 (c). The user's preference model can now be trained with the instance molecule generated.

Converting the RDF Instance Molecule to Machine Learnable data: Since we use a content-based approach for recommendation, we need to convert RDF into a representation that machine learning algorithms can understand. The translation of instance molecules into a table of training data, as used by conventional machine learning algorithms, is relatively straight-forward and is explained below. (Due to lack of space, Fig. 2 (d) only shows a representative set of columns that can be derived from Fig. 2 (c)).

1. **The Table:** All instances belonging to one concept are grouped together in a single table, e.g. banks in Fig. 2 (d).
2. **Rows in the Table:** Each instance molecule to be added to the preference model translates to a row in the table. Instead of dropping the URI, as identifiers usually do not contribute to the learned model, we add the identifier to the table (not depicted in Fig. 2 (d)) to bias the model with a higher preference to a previously visited place.
3. **Columns in the table** The attributes or property-value pairs for the instance get translated as columns and values in the table. The properties for which the type of the table is a domain, appear as columns. For example, while the *hasDriveThruATM* is a property of Banks, the *hasName* property is inherited from the parent concept *POI* and is also present in the table in Fig. 2.
4. **Values in the Table:** RDF Literals in a column are translated as one of string, numeric or nominal values. The relevant information for the conversion into either of these types can be determined from the ranges of the properties in the ontology, or specified explicitly during ontology construction. For example, a value of the *hasName* property translates to string type, the *hasAverageRating* property translates to numeric type and the *hasDriveThruATM* property translates to nominal type. For values of properties that are RDF Blank Nodes, we use nesting of tables where we track inner values for the properties of the blank node (e.g. *hasLocation* property) . For properties with URI values, we can choose to either use the lexical value of the identifier as cell value in the table if we want to introduce bias, or represent the instance in a nested table similar to blank nodes if the values

of its properties are important. For missing attributes, the cells in the table are empty.
5. **Class Column in the Table:** Since visited places translate to only positive training examples, all class values in the table are marked as 'preferred'

Building the Preference Model: The table above, similar to the training data used for a naïve-Bayes classifier, is then used to generate a preference model. The preference model generated is an optimized variation of the table, containing frequency counts/mathematical functions, and helps in calculating the likelihood probabilities. Instead of building the entire preference model from scratch every time a new instance is added, we use an incremental approach.

4 Recommending POIs using a *content-based* approach

In *content-based* recommendation systems, determining if the user likes/dislikes a particular item is a classification problem [11]. A variety of algorithms, like linear regression, rule-based, probabilistic methods, etc. can be used for determining items for recommendation. For example, a naïve-Bayes approach can be used to build a model that can classify a previously unseen POI as *preferred* or *not-preferred*. One can determine which one of n candidate POIs is most likely to be preferred by the user by selecting the POI with the highest (normalized) probability of it being preferred ($P(preferred|POI_i)$). This item can then be recommended to the user. Since the driver history that was collected only contains positive training examples, most of the algorithms mentioned above cannot be used as-is (e.g. in a naïve-Bayes approach, the likelihood and evidence probabilities cancel each other out, giving an inappropriate probability score). In the absence of negative training examples (i.e. places that the user dislikes), we use a nearest-neighbor approach for recommendation. Specifically, from n candidates, which may have been previously unseen, we recommend the item that best matches the ones in the user's preference model. Our approach is described below.

Fetching Candidate POIs and Adding Context: When the user wants to search for POIs, Supe first retrieves candidate places that match the search criteria from the POI Database/Web Services. Necessary user and situation context are added to the POIs after lifting them into RDF, similar to the steps described in Section 3. These POIs can then be scored to find out how likely is the user to prefer each of them.

Nearest-neighbor to a hypothetical Bliss-point Our approach is inspired from the nearest-neighbor algorithms used in clustering and classification.To find how likely is the user to prefer each POI, we first find out how likely is the user to prefer its properties. Let's suppose that each candidate POI has only two properties - name and average rating of all users. The first candidate POI *hasName* "Bank of America" and *hasAverageRating* of 3.5. We can calculate the likelihood probabilities - $P(hasName = \text{“}BankofAmerica\text{”}|preferred)$ and $P(hasAverageRating = 3.5|preferred)$ using the table from Fig. 2, similar to the likelihood calculation in a naïve-Bayes approach. We can plot this point in

a Euclidean space with the properties as the different axes (see Fig. 4 (a)). For a hypothetical POI that is *always* preferred by the user, each of its attributes would have the preference likelihood as 1.0. We can now plot this 'bliss-point' on the Euclidean space. We can similarly plot other candidate POIs (see Fig. 4 (b)). The euclidean distance from the bliss-point is used for the recommendation, and the POI that is nearest to the bliss-point is recommended to the user.

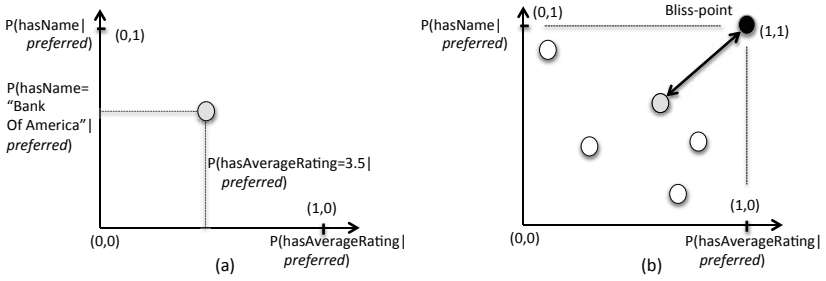


Fig. 4. Plotting candidate POIs using the likelihood probabilities of their attributes. (a) Plotting one candidate. (b) Recommending the nearest neighbor to the Bliss-point.

Distance from Bliss-point We can easily extend the above method to an n-dimensional space. The values of the attributes in the columns in the table in Fig. 2 (d) can either be Literals, Blank Nodes, URIs or missing. For literal values, the likelihood probability is calculated as follows: (i) a Gaussian distribution is used for the probabilities of numeric values (ii) a document similarity metric is used for the probabilities of string values and (iii) symbol probability is used for the probabilities nominal values. The distance for a property with a blank node is calculated recursively, by first calculating the likelihood probabilities of its inner values, and then its distance from another hypothetical bliss-point in its own high dimensional space. For URIs, we can use a dual strategy depending on the nature of the property. In some cases, to bias toward previously seen values for properties, we calculate likelihood as its probability of the occurrence of that URI in that column. If the attributes of the corresponding instance are more important, then the likelihood can be calculated similar to the blank node. The distances from the bliss-point of multiple POIsneed to be normalized before they can be compared. This is done by dividing the distance by the distance of the origin to the bliss-point, thus taking care of missing attributes. For a multivalued property, we take the average of the distances of all its values. As an example, the distance from bliss-point for the Bank of America POI in Fig. 2 (d), would be calculated as follows.

$$D(BofA\_94086) = \sqrt{\frac{\begin{array}{r}(1 - P(\text{“}Bank\ of\ America\text{”} \mid preferred))^2 \\ +(1 - P(2353 \mid preferred))^2 + (1 - P(TRUE \mid preferred))^2 \\ +(1 - P(4.5 \mid preferred))^2 + D(\_ : x)^2\end{array}}{5}}$$

$$Where\ D(\_ : x) = \sqrt{\frac{(1 - P(\text{“}Sunnyvale\text{”} \mid preferred))^2 + (1 - P(\text{“}123\ Murphy\ St.\text{”} \mid preferred))^2}{2}}$$

Recommending the POIs to the User The candidate items retrieved from the database are scored using the *distance from bliss-point* metric and sorted according to the distance. The sorted list of POIs is then sent by the Supe system to the navigation application as a recommendation. Once the user selects a POI from the list, it is fed back to the preference model and the system is able to learn incrementally.

5 The Supe Semantic Web Knowledge System

Since the Supe system is described in detail elsewhere [8], we only describe an overview (see Fig. 5). Supe is a Semantic Web Knowledge System used to collect driver preferences and apply the preference model to provide personalized POI search results to the driver. It is based in the Cloud and contains a Knowledge Base, Intelligent Services, RESTful endpoints and access control mechanisms. To be successful in modeling driver preferences, it needs driver as well as POI data. This semantic data is grounded in an ontology and represented as Linked Data in the Knowledge Base. Supe also provides Intelligent Services, associated with the machine learning task described in the previous sections, for learning the driver's preferences and finding the recommended POIs for the driver. These are wrapped by thin RESTful services that are accessible to the navigation application running on the head-unit and other connected devices for searching POIs and pushing POIs to the navigation application via the Cloud. To prevent RESTful services, Intelligent Services and applications that are running on the connected devices from corrupting other services' or users' data, Supe has an access control mechanism in place. Applications and users use an identifier and a secret passkey combination for authentication. The authorization mechanism is tied to a hierarchical namespace scheme for URIs that governs the policies for data ownership. The RESTful services and Linked Data (URIs starting with 'https://') are accessible on the connected devices side using secured (HTTPS) communication, thus ensuring confidentiality.

6 User Study & Evaluation

We implemented a smart-phone application emulating the in-vehicle navigation application for a user study. The application allowed users to search for POIs belonging to different categories (e.g. gas stations, banks, restaurants, etc.). A cloud-based server supporting the application was implemented using the description of Supe above. Around 50 people, who used their car for running household chores or for their daily commute to work, in the Bay Area (around San Francisco, CA) were selected for the user study. Each user was asked to add at least 10 places to his preference model by driving to the POIs he would visit in

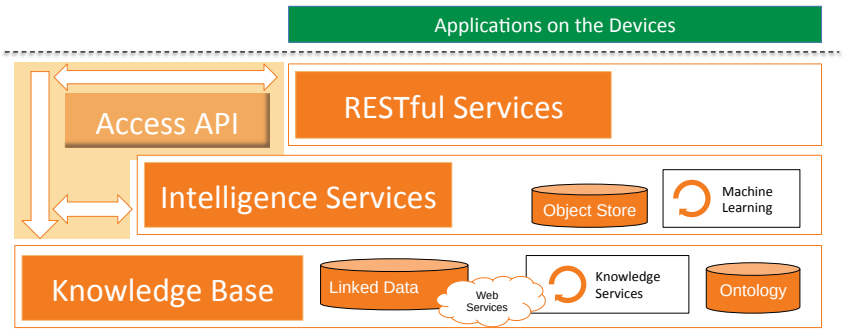


Fig. 5. Semantic Web Knowledge System Layer Cake

his daily life and letting the application know that he like the place, by clicking a 'like' button. Each user was also asked to perform 10 search tasks, where he would search for POIs and choose to one out of the recommended places. The application usage was recorded.

The following screenshot (Fig. 6) shows two instances of POI search for one such user. Initially, there are 3 slots for displaying recommendations, and any other places in the recommended POI list are scrollable. The first image shows the results for search for banks in the user's daily commute area. The ones marked with a pin are the places that he has visited (and liked) and are used to build his preference model. When the user was in a remote place and in need of cash, he searched for banks using the application. The second capture shows that two of his preferred banks were ranked among the top three in the list. Intuitively the reader may figure out that this user prefers a specific banking company. The model is able to detect this preference because its likelihood probability on the *hasName* axis is high. Similar patterns were also detected for other users in different categories, like gas stations, restaurants, etc.

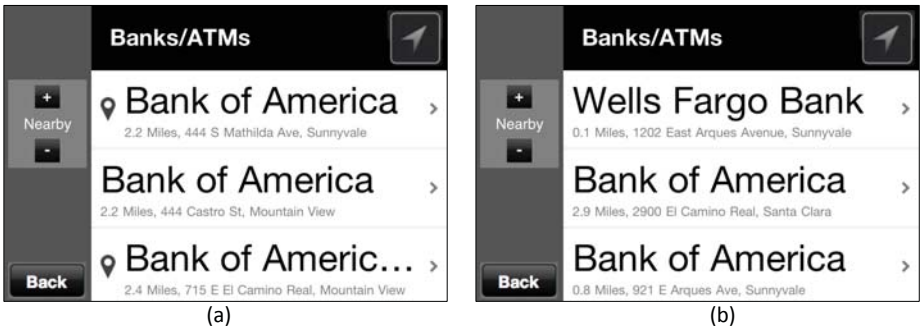


Fig. 6. Screenshots of the Implemented User Study Application: Search result for banks (a) in daily commute area (visited places marked) (b) in an unfamiliar area (using distance from bliss-point metric).

The users had been given two weeks to complete the tasks during which observations were made on the performance of their preference model. For each POI search task, a successful recommendation was counted if, from the 20 candidate POIs that the database returned, the user selected one of the top 3 recommendations. We tracked the success-rate (percentage of successful recommendations in the search task) of the preference model with the increasing size of the preference model and also with the number of searches performed. The selected POI was fed back to the preference model for learning the preference incrementally. Fig. 7 shows the performance of the preference model for one user. As can be seen, after the initial stabilization period, the success rate steadily improved with increasing number of instances in the preference model and as more searches were performed. Overall, for the 48 participants that completed the task, the success rate at the completion of the tasks was 47.63% on an average. This is better than a strategy for random recommendation of 3 out of 20 candidate POIs, which would result in a 15% success rate. Though the improvement in the success-rate was satisfactory, there is a clear scope for better performance. In the future, we intend to use this result as a baseline for comparisons with updates to the learning technique.

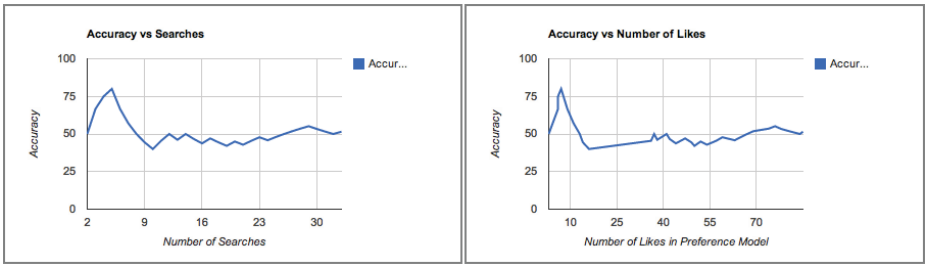


Fig. 7. Performance of the preference model for recommendations made to one user

The use of a probabilistic approach also allowed us to 'look inside' the preference model of a user. For example, for the user in Fig. 6, his preferences for the *hasName* attribute (word cloud) and *distance from current location* context (distribution) are shown in Fig. 8. These visualizations were useful for intuitively understanding the likelihood probabilities of the properties.

7 Related Work

Content-based preference modeling has received large research attention in recommendation systems over the past few years[11]. These algorithms, use machine learning techniques like linear regression, naïve-Bayes, etc. for finding recommendations. One of the earliest works, the Syskill & Webert system[10] uses a naïve-Bayes classifier for classifying web sites as either 'hot' or 'cold'. Similar to our approach, this system also uses the probability score to rank pages according to user's preferences. More recently, personalized information retrieval has also

Fig. 8. Performance of the preference model for Bank recommendations. (a) Word cloud for the *hasName* property for banks, showing user's preference for a banking company over others. (b) Distribution of the values for the *hasDistance* property showing user's preference for banks averaging about 1.4 km away.

been gaining traction in the Semantic Web. For example, *dbrec*[9] is a music recommendation system based on DBpedia that uses a semantic link distance metric for its recommendations. An important constraint that restricted our use of more sophisticated algorithms was the absence of any negative training examples. Incidentally, one-class classification approaches have also been studied, which may present an interesting alternative to our method[12, 5]. In these, clustering, kernel based, or other methods are used to identify the boundary of the class and then predict if an item belongs to that class or is an outlier. The metric used for determining outliers can possibly be used as an alternative to the *distance from bliss-point* metric.

In the past year, the combination of RDF and machine learning has gained some traction. The work that is perhaps most similar to our approach on converting RDF to a machine learning table, is found in Lin et. al [6]. For the movie domain, they try to predict if a movie receives more than $2M in its opening week by converting the RDF graph for the movie to a Relational Bayesian Classifier. One major difference in their approach to ours is the translation of multi-valued object properties (e.g. *hasActor*) into the relational table. While doing so, their approach 'flattens' all objects. For example, names of all actors get aggregated into a single 'has actor name' column and all actors' year of birth get aggregated into a 'has actor YoB' column. The associativity within an instance (e.g. of an actor's name to his age) is lost. In our approach, this advantage, of property associativity using a graph, is maintained since we determine the score for each blank node independently. However, a more in-depth comparison of the two approaches needs to be conducted on common data for better analysis. Another related work in learning from RDF has been explored in Bicer et. al[3], where relational kernel machines are used for movie recommendations. A similar kernel based approach was also used in Lösch et. al [7] for recommending new links in RDF graphs - for example, recommending known people (*foaf:knows*).

8 Conclusion & Future Work

In this paper we described a semantic content-based approach for recommending POIs according to driver preferences learned using the navigation history.

Our approach was able to build the preference model using RDF data associated with the driver history as a result of the following: (i) it used an easy translation method between semantic content (RDF data) into machine learnable tables (ii) the content-based approach was easily extendable to include and learn the contextual preference (e.g. *distanceFromWork*, etc.) (iii) the *distance from bliss-point* metric was used to recommend POIs from a set of candidates using the modeled preferences (iv) the probabilistic nature of the likelihood of the attributes in the preference model helped study the user's preferences by visualization. This was verified through a user study that produced a 47.63% success-rate.

Though our preliminary evaluation shows promising results with actual users, using simple metrics, it has scope for improvements. We intend to explore other machine learning techniques (e.g. kernel based, one-class classification, etc.) to improve the preference model, as future work. We also intend to test this on a larger dataset of driver history.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on 17(6), 734–749 (2005)
2. Almazro, D., Shahatah, G., Albdulkarim, L., Kherees, M., Martinez, R., Nzoukou, W.: A survey paper on recommender systems. Arxiv preprint arXiv:1006.5278 (2010)
3. Bicer, V., Tran, T., Gossen, A.: Relational kernel machines for learning from graph-structured rdf data. The Semantic Web: Research and Applications pp. 47–62 (2011)
4. Ding, L., Finin, T., Peng, Y., Da Silva, P., McGuinness, D.: Tracking rdf graph provenance using rdf molecules. In: Proc. of the 4th International Semantic Web Conference (Poster) (2005)
5. Khan, S., Madden, M.: A survey of recent trends in one class classification. Artificial Intelligence and Cognitive Science pp. 188–197 (2010)
6. Lin, H., Koul, N., Honavar, V.: Learning relational bayesian classifiers from rdf data. The Semantic Web–ISWC 2011 pp. 389–404 (2011)
7. Lösch, U., Bloehdorn, S., Rettinger, A.: Graph kernels for rdf data. The Semantic Web: Research and Applications pp. 134–148 (2012)
8. Parundekar, R., Oguchi, K.: Learning driver preferences of pois using a semantic web knowledge system. The Semantic Web: Research and Applications pp. 703–717 (2012)
9. Passant, A.: Dbrec, music recommendations using dbpedia. The Semantic Web–ISWC 2010 pp. 209–224 (2010)
10. Pazzani, M., Billsus, D.: Learning and revising user profiles: The identification of interesting web sites. Machine learning 27(3), 313–331 (1997)
11. Pazzani, M., Billsus, D.: Content-based recommendation systems. The adaptive web pp. 325–341 (2007)
12. Tax, D.: One-class classification. PhD thesis, Delft University of Technology (June 2001), `http://www.ph.tn.tudelft.nl/davidt/thesis.pdf`

Semantic Network-driven News Recommender Systems: a Celebrity Gossip Use Case

Marco Fossati, Claudio Giuliano, and Giovanni Tummarello
{fossati,giuliano,tummarello}@fbk.eu

Fondazione Bruno Kessler, via Sommarive 18, 38123 Trento, Italy

Abstract. Information overload on the Internet motivates the need for filtering tools. Recommender systems play a significant role in such a scenario, as they provide automatically generated suggestions. In this paper, we propose a novel recommendation approach, based on semantic networks exploration. Given a set of celebrity gossip news articles, our systems leverage both natural language processing text annotation techniques and knowledge bases. Hence, real-world entities detection and cross-document entity relations discovery are enabled. The recommendations are enhanced by detailed explanations to attract end users' attention. An online evaluation with paid workers from crowdsourcing services proves the effectiveness of our approach.

Keywords: Data Integration, Natural Language Processing, Information Retrieval, Information Filtering, Entity Linking, Recommendation Strategy

1 Introduction

The amount of publicly available data on the World Wide Web nowadays has dramatically increased and has led to the problem of information overload. Recommender systems try to tackle this issue by offering personalized suggestions. News recommendation is a real-world application of such systems and is growing as fast as the online news reading practice: it is estimated that, in May 2010, 57% of U.S. Internet users consumed online news by visiting news portals [7]. Recently, online news consumers seem to have changed the way they access news portals: "just a few years ago, most people arrived at our site by typing in the website address. (...) Today the picture is very different. Fewer than 50% of the 8 million+ visitors to the News website every day see our front page and the rest arrive directly at a story", a product manager of the BBC News website affirms,[1] indicating the need for news information filtering tools.

The online reading practice leads to the so-called *post-click* news recommendation problem: when a user has clicked on a news link and is reading an article, he or she is likely to be interested in other related articles. This is still a typical editor's task, namely an expert who manually looks for relevant content and

[1] http://www.bbc.co.uk/blogs/bbcinternet/2012/03/bbc_news_facebook_app.html

builds a recommendation set of links, which will be displayed below or next to the current article. The primary aim is to keep users navigating on the visited portal. News recommender systems attempt to automate such task. Current strategies can be clustered into 3 main categories [5], namely (a) collaborative filtering, (b) content-based recommendation, and (c) knowledge-based recommendation. (a) focuses on the similarities between users of a service, thus relying on user profiles data. (b) leverages term-driven information retrieval techniques to compute similarities between items. (c) mines external data to enrich item descriptions.

In this paper, we propose a novel news recommendation strategy, which leverages both natural language processing techniques and semantically structured data. We show that entity linking tools can be coupled to existing knowledge bases in order to compute unexpected suggestions. Such knowledge bases are used to discover meaningful relations between entities. As a preliminary work to assess the validity of our approach, we focus on a celebrity gossip use case and consume data from the TMZ news portal and the Freebase graph database.[2] For instance, given a TMZ article on `Michael Jackson`, our strategy is able to detect from Freebase that `Michael Jackson` (a) is a dead celebrity who had drug problems and (b) dated with `Brooke Shields`, thus suggesting other TMZ articles on `Amy Winehouse`, `Kurt Cobain` (other dead celebrities who had drug problems) and `Brooke Shields`. We investigate if user attention can be attracted via specific explanations, which clarify why a given recommendation set is proposed. Such explanations are built on top of the entity relations. Finally, we conducted an online evaluation with real users. We outsourced a set of experiments to the community of paid workers from Amazon's Mechanical Turk (AMT) crowdsourcing service.[3] The collected results confirm the effectiveness of our approach.

Our primary aim is to attract the attention of a generic user, since post-click news recommendation generally relies on a single click user profile data. Therefore, we are set apart from most traditional recommender systems with respect to three main features:

1. *User agnosticity*: user interests are deduced from user profile data and contribute to the quality of recommendations. Collecting explicit feedback is a costly task, as it requires motivated users. Our approach gives low priority to user profiles.
2. *Unexpectedness*: similarity, novelty and coherence are key components for satisfactory news recommendations [7]. Content-based strategies tend to propose too similar items and create an 'already seen' sensation. We believe entity relations discovery can augment both novelty and coherence, thus leading to unexpected suggestions.
3. *Specific explanation*: in news web portals, generic sentences such as `Related stories` or `See also` are typically shown together with the recommendation set. We expect that more specific sentences can improve the trustworthiness of the system.

[2] `http://www.tmz.com`, `http://www.freebase.com/`
[3] `https://www.mturk.com/mturk/welcome`

2 Related Work

Content-based recommendation applies to unstructured text, such as news articles. Document representation with bag-of-words vector space models and the cosine similarity function still represent a valid starting point to suggest topic-related documents [11]. Knowledge extraction from structured data is an attested knowledge-based strategy. Linked Open Data (LOD) datasets, e.g., DBpedia[4] and Freebase are queried to enrich with properties the entities extracted from news articles [6], to collect movie information for movie schedules recommendations [12], or to suggest music for photo albums [1]. Structured data may be also mined in order to compute similarities between items, then between user and items [5]. Content-based and knowledge-based approaches must be combined into hybrid systems in order to achieve better results. Lašek [6] proposes a hybrid news articles recommendation system, which merges content processing techniques and data enrichment via LOD.

Recommender systems evaluation frameworks boil down to two main approaches [5], namely (a) offline and (b) online. (a) leverages gold-standard datasets and aims at estimating the performance of a recommendation algorithm via statistical measures. (b) relies on real user studies. Ziegler et al. [13] adopt both approaches. Hayes et al. [4] argue that user satisfaction corresponds to the actual use of a system and can be effectively measured only via online evaluation. The interest in exploiting crowdsourcing services for dataset building and online evaluation has recently grown, especially with respect to natural language processing tasks [10] and behavioral research [8].

3 Approach

Our strategy merges content-based and knowledge-based approaches and is defined as a *hybrid entity-oriented* recommendation strategy enhanced by human-readable explanations. Given a source article from a news portal, we recommend other articles from the portal archive, namely the corpus, by leveraging both entity linking techniques and knowledge extraction from semantically structured knowledge bases. Specifically, we gathered a celebrity gossip corpus from TMZ and chose Freebase as the knowledge base.

We consider both the corpus and the knowledge base as a unique object, namely a *dataspace*, which results from heterogeneous data sources integration. Each data source is converted into an RDF graph and becomes an element of the dataspace. Such dataspace can then be queried in order to retrieve sets of recommendations. A *semantic recommender* exploits SPARQL graph navigation capabilities to output recommendation sets. Each recommender is built on top of a concept, e.g., *substance abuse*.

The entity linking step in the corpus processing phase enables the detection of both real-world entities and encyclopedic concepts. We compute concept statistics on the whole corpus and assume that the most frequent ones are likely to generate interesting recommendations. A mapping between corpus concepts and meaningful relations of the knowledge base allows the creation of recom-

[4] `http://dbpedia.org/`

menders. Table 1 shows the TMZ-to-Freebase n-ary concept mapping we manually built. Each Freebase value represents the starting point for the construction of a recommender, while the string after the last dot becomes the name of the recommender, e.g., *parents*.

Given an entity of the source article, a name of a recommender and an entity contained in the recommendation sets, we are able to construct a specific explanation. Ultimately, a ranking of all the recommendation sets produces the final top-N suggestions output.

Table 1: TMZ-to-Freebase mapping

TMZ	Freebase
Family	people.person.{parents, sibling_s, children, spouse_s}
Intimate_relationship	celebrities.celebrity.sexual_relationships
Dating	base.popstra.celebrity.dated
Ex_(relationship)	base.popstra.celebrity.breakup
Net_worth	celebrities.celebrity.net_worth
Substance_abuse	celebrities.celebrity.substance_abuse_problems
Conviction	base.crime.convicted_criminal
Court	law.court.legal_cases
Arrest	base.popstra.celebrity.{arrest, prison_time}
Legal_case	law.legal_case.subject
Criminal_charge	celebrities.celebrity.legal_entanglements
Judge	law.judge
Death	people.deceased_person
Television_program	tv.tv_program

4 System Architecture

Figure 1 describes the general system workflow. The major phases are (a) corpus processing, (b) knowledge base processing, (c) dataspace querying and (d) recommendation ranking.

TMZ Processing Pipeline. Given as input a set of TMZ articles, we output an RDF graph and load it into the dataspace. Corpus documents are harvested via a subscription to the TMZ RSS feed. The RSS feed returns semi-structured XML documents. A cleansing script extracts raw text from each XML document. The entity linking step exploits *The Wiki Machine*,[5] a state-of-the-art [9] machine learning system designed for linking text to Wikipedia, based on a word sense disambiguation algorithm [2]. For each raw text document, real-world entities such as persons, locations and organizations are recognized, as well as encyclopedic concepts. This enables (a) the assignment of a unique identifier, namely a DBpedia URI to each annotation and (b) the choice of top corpus concepts for recommenders building purposes. The Wiki Machine takes a plain text as input and produces an RDFa document.[6] The extracted terms are assigned an `rdf:type`, namely `NAM` for real-world entities or `NOM` for encyclopedic concepts. The `hasLink` property connects the terms to the article URL they belong, thus enabling the computation of the recommendation set. Other metadata, such as

[5] `http://thewikimachine.fbk.eu`

[6] The full corpus of TMZ RDFa documents is available at `http://bit.ly/QLph9B`

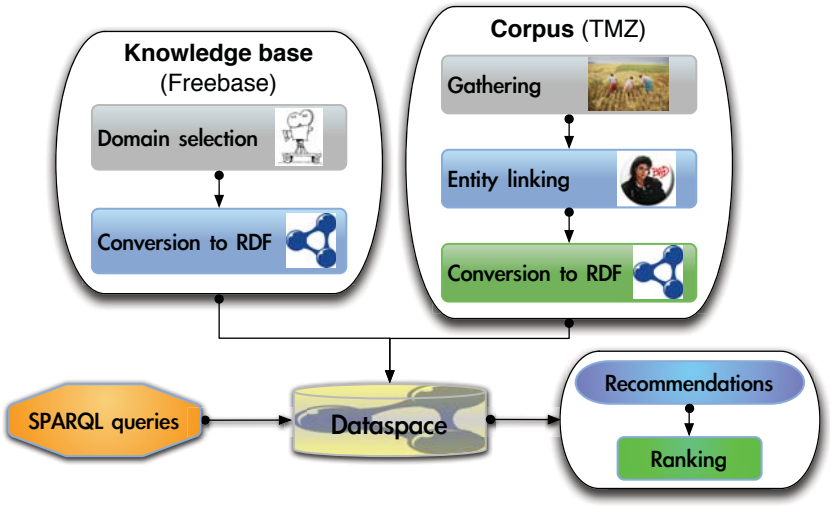


Fig. 1: High level system workflow

the link to the corresponding Wikipedia page and the annotation confidence score are also expressed. RDFa documents are converted into RDF data via the Any23 library.[7] RDF data is loaded into a Virtuoso[8] triple store instance, which serves the dataspace for querying.

Freebase Processing Pipeline. Freebase provides exhaustive granularity for several domains, especially for celebrities. Given that such knowledge base is large, we avoid loading its complete version, because of severe performance issues we encountered. Consequently, meaningful slices corresponding to the corpus domains, e.g., celebrities, people, are selected. A domain-dependent subset is then produced via a filter written in Java. The dataset is converted into RDF data with logic implemented in Java. Finally, RDF data is loaded into a Virtuoso triple store instance.

4.1 Querying the Dataspace

A recommender performs a join between an entity belonging to the TMZ graph and the corresponding entity belonging to the Freebase graph. TMZ entities are identified by a DBpedia URI, which differs from the Freebase one. Therefore, we exploit `sameAs` links between DBpedia and Freebase URIs. Recommenders are divided in two categories, namely (a) *entity-driven* and (b) *property-driven*.[9] For each detected entity of the source article, we run Freebase schema inspection queries[10] and retrieve its types and properties. Thus, we are able to recognize which recommenders can be triggered for a given entity. Building a recommender

[7] `http://incubator.apache.org/any23/`
[8] `http://virtuoso.openlinksw.com/`
[9] The full sets are available at `http://bit.ly/MWGu06` and `http://bit.ly/MWGsW3`
[10] Available at `http://bit.ly/MVGVtE`

requires (a) knowledge of relevant Freebase schema parts in order to properly browse its graph and (b) a sufficiently expressive RDFa model for named entities and link retrieval. The `NAM` type and the `hasLink` property provide such expressivity.

Entity-Driven Recommenders. The queries behind entity-driven recommenders contain an `%entity%` parameter that must be programmatically filled by an entity belonging to the source article. For instance, given an article in which `Jessica Simpson` is detected and triggers the *sexual relationships* recommender, we are able to return all the corpus articles (if any) that mention entities who had sexual relationships with her, e.g., `John Mayer`. To avoid running empty-result recommenders, we built a set of `ASK` queries,[11] which check if recommendation data exists for a given entity. The *sexual relationships* query follows:

```
PREFIX fb: <http://rdf.freebase.com/ns/>
PREFIX twm: <http://thewikimachine.fbk.eu#>
SELECT DISTINCT ?had_relationship_with ?link
WHERE  <http://dbpedia.org/resource/%entity%> owl:sameAs ?fb_entity .
?fb_entity fb:celebrities.celebrity.sexual_relationships ?fb_sexual_rel .
?fb_sexual_rel fb:celebrities.romantic_relationship.celebrity ?fb_celeb .
?fb_celeb fb:type.object.name ?had_relationship_with .
?dbp_celeb owl:sameAs ?fb_celeb ; a twm:NAM ; twm:hasLink ?link ; twm:hasConfidence ?conf .
FILTER (?fb_entity != ?fb_celeb) . FILTER (lang(?had_relationship_with)='en') .
ORDER BY DESC (?conf)
```

Property-Driven Recommenders. After the schema inspection step, an entity of the source article can directly trigger one of these recommenders if it contains the corresponding property. Property-driven queries return articles that mention entities who share the same property. Hence, they do not require a parameter to be filled. For instance, given an article in which `Lindsay Lohan` is detected and the property `legal entanglements` is identified during the schema inspection step, we can suggest other articles on people who had legal entanglements, e.g., `Britney Spears`.

Building Explanations. Specific explanations are handcrafted from <*s, r, o*> triples, where *s* is a *subject* entity that was extracted from the source article, *r* is the *relation* expressed by the triggered recommender and *o* is an *object* entity for which the recommendation set is computed. Therefore, we are able to construct different explanations depending on the elements we use. For instance, (a) *s,r,o* yields: `Jessica Simpson` had `sexual relationships` with `John Mayer`. Read more about him. (b) *s,r* yields: Read more about `Jessica Simpson`'s `sexual relationships`. (c) *r,o* yields: Read more about her `sexual relationships` with `John Mayer`.

4.2 Ranking the Recommendation Sets

Since recommendations originate from database queries, they are unranked and in some cases too many. To overcome the problem, we implemented an information retrieval ranking algorithm and are able to provide top-N recommendations.

[11] Available at `http://bit.ly/NDNORH`

The bag-of-words (BOW) cosine similarity function is known to perform effectively for topic-related suggestions [11]. However, it does not take into account language variability. Consequently, we also leverage a latent semantic analysis (LSA) algorithm.[12] The final score of each corpus article is the sum of BOW and LSA scores and is assigned to the article URL. Afterwards, we run all the recommenders and intersect their result sets with the BOW+LSA ranking of the whole corpus, thus producing a so-called *semantic* ranking. This represents our final output, which consists of a ranked set of article URLs associated to the corresponding recommenders names.

5 Evaluation

The assessment of end user satisfaction has high priority in our work. According to Hayes et al. [4], we consequently decided to adopt an online evaluation approach with real users. In this scenario, the major issue consists of gathering a sufficiently large group of people who are willing to evaluate our systems. Crowdsourcing services provide a solution to the problem, as they allow us to outsource the evaluation task to an already available massive community of paid workers. To the best of our knowledge, no news recommender systems have been evaluated with crowdsourcing services so far. We set up an experimental evaluation framework for AMT, via the CrowdFlower platform.[13] A description of the mechanisms that regulate AMT is beyond the scope of the present paper: the reader may refer to [8] for a detailed analysis.

Our primary aim is to demonstrate that evaluators generally prefer our recommendations. Thus, we need to put our strategy in competition with a baseline. We leveraged the already implemented BOW+LSA information retrieval ranking algorithm. In addition, we set two specific objectives, related to the *specific explanation* and *unexpectedness* assumptions, as outlined in Section 1: (a) confirm that a specific explanation better attracts user attention rather than a generic one; (b) check if the recommended items are interesting, although they may appear unrelated and no matter what kind of explanation is provided.

Quality control of the collected judgements is a key factor for the success of the experiments. The essential drawback of crowdsourcing services relies on the cheating risk: workers (from now on called *turkers*) are generally paid a few cents for tasks which may only need a single click to be completed. Hence, it is highly probable to collect data coming from random choices that can heavily pollute the results. The issue is resolved by adding *gold* units, namely data for which the requester already knows the answer. If a turker misses too many gold answers within a given threshold, he or she will be flagged as untrusted and his or her judgments will be automatically discarded.

5.1 General Setting

Our evaluation framework is designed as follows: (a) the turker is invited to read a complete news article. (b) A set of recommender systems are displayed

[12] `http://hlt.fbk.eu/en/technology/jlsi`
[13] `http://crowdflower.com/`

below the article. Each system consists of a natural language *explanation* and a news title *recommendation*. (c) The turker is asked to give a preference on the most attracting recommendation, namely the one he or she would click on in order to read the suggested article. A single experiment (or *job*) is composed of multiple data *units*. A unit contains the text of the article and the set of explanation-recommendation pairs. Figure 2 shows a unit fragment of the actual web page that is given to a turker who accepted one of our evaluation jobs. Both instructions and question texts need to be carefully modeled, as they must mirror the main objective of the task and should not bias turkers' reaction. Since we aim at evaluating user attention attraction, we formulated them as per Figure 2.

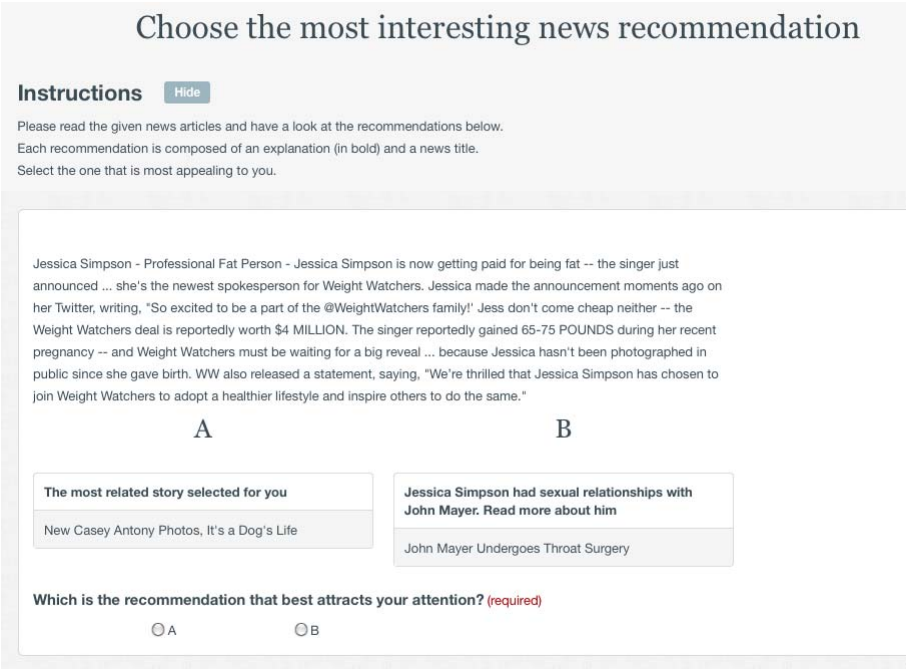


Fig. 2: Web interface of an evaluation job unit

5.2 Experiments

Table 2 provides an overview of our experimental environment. The parameters we have isolated for a single experiment are presented in Table 2a. On top of the possible variations, we built a set of nine experiments, which are described in Table 2b. We modeled two Q values, namely direct (as per Figure 2) and indirect (`Which recommendation do you consider to be more trustworthy?`), to monitor a possible alteration of turkers' reaction. Experiments having $A = 5$ aim at decreasing the probability a turker gets trusted by chance, because he or she accidentally selected correct gold answers. They have an additional F value in the *Rec* parameter, as we randomly extracted 3 fake recommendations per

unit from a file with more than 2 million news titles. However, such an architectural choice generated noisy results, since it occurred that some fake titles were selected.[14] *Exp* is a key parameter, which allows us to check whether the presence or the absence of a specific explanation represents a discriminating factor. *SExp* is intended to measure the effectiveness of a specific explanation while reducing its complexity.

Table 2: Experiments overview

(a) Parameters

Parameter	Values
Q	D, I
A	B, M
Exp	GS, G
SExp	SRO, SR, RO, R
Rec	B, S, F

(b) Configuration

Name	Q	A	Exp	SExp	Rec
Pilot	D	2	GS	SRO	B, S
Same explanation	D	2	G	None	B, S
4 generic + 1 specific	D	5	GS	SRO	B, S, F
5 generic	D	5	G	None	B, S, F
Same recommendation	D	2	GS	SRO	S
Relation only	D	5	GS	R	B, S, F
Subject + relation	D	5	GS	SR	B, S, F
Object + relation	D	5	GS	RO	B, S, F
Indirect	I	2	GS	SRO	B, S

Legend

Q	Question
A	Answer
Exp	Explanation
SExp	specific explanation
Rec	Recommendation
D	Direct
I	Indirect
2	Binary
5	5 choices
GS	Generic + specific
G	Generic only
SRO	Subject + relation + object
SR	Subject + relation
RO	Relation + object
R	Relation only
B	Baseline
S	Semantic
F	Fake

Each job contains 8 regular + 2 gold units, namely 5 articles proposed twice, in combination with 2 significant (and eventually 3 fake) explanation-recommendation pairs. The recommendation titles of the regular units are extracted from the top-2 links of the baseline and the semantic rankings. Gold is created by extracting the title from the last, i.e., less related link of the baseline ranking, the top link of the semantic ranking and assigning the correct answer to the latter. We collected a minimum of 10 valid judgments per unit and set the number of units per page to 3.

Once the results obtained, it frequently occurred that the expected number of judgments was higher: depending on their accuracy in providing answers to gold units, turkers switched from untrusted to trusted, thus adding free extra judgments. The proposed articles come from the TMZ website, which is well known in the United States. Therefore, we decided to gather evaluation data only from American turkers. The total cost of each experiment was 3.66$.

After visiting some news web portals, we chose the following generic explanations and randomly assigned them to both the baseline and the fake recommendations: (a) `The most related story selected for you`; (b) `If you liked`

[14] See Table 3 for further details.

Table 3: Absolute results per experiment. ♢, ♠ and ♣ respectively indicate statistical significance differences between baseline and semantic methods, with $p < 0.05$, $p < 0.01$ and $p < 0.001$

Experiment	Judgments	Fake %	Baseline %	Semantic %
Pilot 1	82	0	40.24	**59.76**♢
Pilot 2	80	0	32.5	**67.5**♠
Same explanation	80	0	48.75	51.25
4 generic + 1 specific	90	3.33	23.33	**73.33**♣
5 generic	88	13.63	37.5	48.86
Same recommendation	86	0	36.04	**63.96**♠
Relation only	68	13.23	41.17	45.58
Indirect	82	0	37.8	**62.2**♠
Subject + relation	86	8.13	41.86	50
Object + relation	68	5.88	41.17	52.94

`this article, you may also like`; (c) `Here for you the hottest story from a similar topic`; (d) `More on this story`; (e) `People who read this article, also read`. 2 regular units were removed from the *relation only* and the *object + relation* experiments: it was impossible to build specific explanations with an implicit subject or object, since the entities that triggered the recommendations differed from the main entity of the source article.

5.3 Results

Table 3 provides an aggregated view of the results obtained from the Crowdflower platform.[15] With respect to the absolute percentage values, we first observe that our approach always outperformed the baseline. Furthermore, statistical significance differences emerge when a complete $<s, r, o>$ specific explanation is given. We ran twice, i.e., in two separate days the *pilot* experiment and noticed an improvement. The *indirect* experiment only differs from the pilot in the question parameter and yielded similar results. The *4 generic + 1 specific* experiment has the highest semantic percentage: this translates into an expected behavior, since the presence of a single specific explanation against four generic ones is likely to bias turkers' reaction towards our approach. As the complexity of the specific explanation decreases, i.e., in the *subject + relation, object + relation* and *relation only* experiments or when only generic explanations are presented, namely in the *5 generic* and *same explanation* experiments, judgments towards our approach tend to decrease too. Hence, we evince the importance of providing specific explanations in order to attract user attention.

5.4 Discussion

Experiments containing a specific explanation aim at assessing its attractive power (assumption 3). If we compare experiments which only differ in the *Exp* parameter, namely *4 generic + 1 specific* and *5 generic*, *pilot 1-2* and *Same explanation*, in the formers turkers prefer our strategy with a statistically sig-

[15] The complete set of full reports is available at `http://bit.ly/MOrN30`

nificant difference. Therefore, specific explanations are proven to enhance the trustworthiness of the system.

The evaluation of the unexpectedness factor (assumption 2) boils down to check whether turkers privilege the novelty of a recommendation or its similarity to the source article. In experiments including only generic explanations, namely *Same explanation* and *5 generic*, we noticed the following: (a) no statistically significant differences exist between the strategies; (b) when the baseline returns articles that are unrelated to the topic or the entity of the source article, turkers prefer our strategy and vice versa. Hence, we argue that users tend to privilege similarity if they are given a generic explanation. On the other hand, when the baseline strategy suggests a clearly related article and when a specific explanation is provided, turkers tend to choose our strategy even if it suggests an apparently unrelated article. This is a first proof of the unexpectedness factor: users are attracted by the specific explanation and are eager to read an unexpected article rather than another article on the same topic/entity.

6 Conclusion

In this paper, we presented a novel recommendation strategy leveraging entity linking techniques in unstructured text and knowledge extraction from structured knowledge bases. On top of it, we build hybrid entity-oriented recommender systems for news filtering and post-click news recommendation. We argued that entity relations discovery leads to unexpected suggestions and specific explanations, thus attracting user attention. The adopted online evaluation approach via crowdsourcing services assessed the validity of our systems. A demo prototype consumes Freebase data to recommend TMZ celebrity gossip articles and can be viewed at `http://spaziodati.eu/widget_recommendation/`. For our future work, we have set the following milestones:

1. *Ecological evaluation.* AMT allowed us to build fast and cheap online evaluation experiments. However, the collected judgments may be biased by the politeness effect of the economical reward and the turkers' awareness of performing a question-answering task. Therefore, we intend to set up an ecological evaluation scenario, which simulates a real-world usage of our recommender systems and enables natural user reactions. We will adopt the Google AdWords[16] approach proposed by Guerini et al. [3].
2. *Methodology for building recommenders.* Currently, we have manually implemented a domain-specific list of recommenders, based on the most frequent corpus concepts. We plan to automate this process by extracting generic relations from Freebase via data analytics techniques.
3. *Methodology for building specific explanations.* Explanations are naively mapped to the relations and the corresponding subject/object entities. How to automatically build linguistically correct sentences remains an open problem.
4. *User profile construction.* Explicit and implicit user preferences acquisition can improve the quality of the recommendations. Our demo page may serve

[16] `http://adwords.google.com/`

as a platform for gathering such data. Otherwise, we may adapt our systems to datasets containing user ratings.

Acknowledgements. This work was supported by the EU project Eurosentiment, contract number 296277.

References

1. Chao, J., Wang, H., Zhou, W., Zhang, W., Yu, Y.: Tunesensor: A semantic-driven music recommendation service for digital photo albums. In: Proceedings of the 10th International Semantic Web Conference. ISWC2011 (October 2011)
2. Giuliano, C., Gliozzo, A.M., Strapparava, C.: Kernel methods for minimally supervised wsd. Computational Linguistics 35(4), 513–528 (2009)
3. Guerini, M., Strapparava, C., Stock, O.: Ecological evaluation of persuasive messages using google adwords. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics. ACL2012, vol. abs/1204.5369 (July 2012)
4. Hayes, C., Cunningham, P., Massa, P.: An on-line evaluation framework for recommender systems. Tech. Rep. TCD-CS-2002-19, Trinity College Dublin, Department of Computer Science (2002)
5. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender Systems: An Introduction. Cambridge University Press (2011)
6. Lašek, I.: Dc proposal: Model for news filtering with named entities. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) The Semantic Web – ISWC 2011, Lecture Notes in Computer Science, vol. 7032, pp. 309–316. Springer Berlin / Heidelberg (2011)
7. Lv, Y., Moon, T., Kolari, P., Zheng, Z., Wang, X., Chang, Y.: Learning to model relatedness for news recommendation. In: Proceedings of the 20th international conference on World wide web. pp. 57–66. WWW '11, ACM, New York, NY, USA (2011)
8. Mason, W., Suri, S.: Conducting behavioral research on amazon's mechanical turk. Behavior Research Methods 44, 1–23 (2012)
9. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: Dbpedia spotlight: shedding light on the web of documents. In: Proceedings of the 7th International Conference on Semantic Systems. pp. 1–8. I-Semantics '11, ACM, New York, NY, USA (2011)
10. Negri, M., Bentivogli, L., Mehdad, Y., Giampiccolo, D., Marchetti, A.: Divide and conquer: crowdsourcing the creation of cross-lingual textual entailment corpora. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 670–679. EMNLP '11, Association for Computational Linguistics, Stroudsburg, PA, USA (2011)
11. Pazzani, M., Billsus, D.: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web, Lecture Notes in Computer Science, vol. 4321, pp. 325–341. Springer Berlin / Heidelberg (2007)
12. Thalhammer, A., Ermilov, T., Nyberg, K., Santoso, A., Domingue, J.: Moviegoer - semantic social recommendations and personalized location-based offers. In: Proceedings of the 10th International Semantic Web Conference. ISWC2011 (October 2011)
13. Ziegler, C.N., Lausen, G., Schmidt-Thieme, L.: Taxonomy-driven computation of product recommendations. In: Proceedings of the thirteenth ACM international conference on Information and knowledge management. pp. 406–415. CIKM '04, ACM, New York, NY, USA (2004)

Cinemappy: a Context-aware Mobile App for Movie Recommendations boosted by DBpedia

Vito Claudio Ostuni[1], Tommaso Di Noia[1], Roberto Mirizzi[2], Davide Romito[1], Eugenio Di Sciascio[1]

[1]Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Italy
{ostuni,d.romito}@deemail.poliba.it, {t.dinoia,disciascio}@poliba.it
[2]HP Laboratories, Palo Alto, CA, USA
roberto.mirizzi@hp.com

Abstract. The recent spread of the so called Web of Data has made available a vast amount of interconnected data, paving the way to a new generation of ubiquitous applications able to exploit the information encoded in it. In this paper we present Cinemappy, a location-based application that computes contextual movie recommendations. Cinemappy refines the recommendation results of a content-based recommender system by exploiting contextual information related to the current spatial and temporal position of the user. The content-based engine leverages DBpedia, one of the best-known datasets publicly available in the Linked Open Data (LOD) project.

1 Introduction

Context can be defined as "*any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves*" [10]. This concept is particularly relevant whenever a user needs to look for information that do not depend exclusively from the particular knowledge domain. Thanks to great technological advances occurred in the latest years, particularly in ubiquitous computing, users are able to run almost any kind of application and to perform almost any task on small mobile devices. Smartphones and tablets are becoming a primary platform for information access [22]. If we think of a recommender task in a mobile scenario (e.g., choosing a movie in one of the nearest movie theaters, planning a sightseeing, etc.), we see that most recommendations are requested by users while they are on their way. This causes a continuous change in the context that needs to be carefully addressed. Recommendations are much more useful and enjoyable for end users as they change with their current context [6].

Recommender systems (RS) are information filtering and decision support tools addressing the problem of information overload, providing product and service recommendations personalized for user's needs and preferences.

In this paper we present an implementation of a content-based **C**ontext-**A**ware **R**ecommender **S**ystem (CARS) that gets information needed to recommend items from Linked Open Data (LOD) datasets [14] and combines it with

other information freely available on the Web. In particular, our system recommends movies to be watched in theaters that are located in the user's neighborhood. The recommendation leverages `DBpedia` [7] as the knowledge base whence we extract movie features, such as genres, actors, directors, subjects, etc.. Main contributions of this paper are: (a) a semantic context-aware recommender system. In our approach we use semantic repositories and the notion of context to provide users with meaningful recommendations in a mobile environment; (b) the exploitation of heterogeneous information sources for movie recommendation. In particular we leverage data coming both from the `Web of Data` and from the traditional Web; (c) a proof-of-concept Android application exposing geo-location and context-aware features for movie recommendations in user neighborhood.

The remainder of this paper is organized as follows. In Section 2 we give some background information about the notion of context in recommender systems and how we exploit it in our approach. In Section 3 we present some basic notions about the usage of `DBpedia` as knowledge base for recommender systems. Then, in Section 4 we detail our approach and we present our mobile app. In Section 5 we discuss relevant related work. Conclusion and future work close the paper.

2 Context-Aware Recommender Systems

A CARS deals with modeling and predicting user preferences by incorporating available contextual information into the recommendation process. Preferences and tastes are usually expressed as ratings and are modeled as a function of items, users and context. Hence, the rating function can be defined as:

$$r : User \times Item \times Context \rightarrow Rating$$

with the obvious meaning for $User$ and $Item$. $Context$ is defined by means of different attributes and criteria (we will detail them in the following) while *Rating* is usually represented as a Likert scale (going from 5 to 10 different values) or as a *Like/Don't like* Boolean set.

As in [19], we assume that there is a predefined finite set of contextual types in a given application and each of these types has a well-defined structure. In particular, in our mobile scenario we consider the context as represented by the following information:

Companion. There are many situations where a place or a service is more or less enjoyable depending on the people that are together with the user. Maybe she and her friends love romantic movies but this is not the case of her husband. So, it would be fine if a movie recommender engine suggested romantic movies when she is with her friends and comedies when she is with her husband.

Time. This is another important feature to consider. For example, in a movie theater recommender system, all the movies scheduled before the current time, plus the time to get to the theatre, have to be discarded.

Geographic relevance. Geo-localized information plays a fundamental role in mobile applications. Depending on the current location of the user, a recommender engine should be able to suggest items close to them and discard the

farther ones even if they may result more appealing with respect to the query. A location-aware recommender system should be able to suggest items or services for a given user whose location is known, considering more useful criteria than simply distance information. In [9] the authors propose ten criteria of geographic relevance. In the following we describe five of them that we considered relevant for our mobile application:

- ***Hierarchy***: it represents the degree of separation between the current position of the user and that of the suggested item within a predefined spatial hierarchy. The main assumption is that geographic units are cognitively and empirically organized into a nested hierarchical form (e.g., city districts).
- ***Cluster***: it is the degree of membership of an entity to a spatial cluster of related or unrelated entities. The user might be more interested in visiting a mall than a single shop.
- ***Co-location***: a user prefers locations where she may find other useful entities co-located with the one representing their main interest. As an example, it is common to have restaurants close to cinemas (since people like to go for dinner before watching or after having watched a movie).
- ***Association Rule***: this criterion represents possible association rules that relates an entity with a related collection of geographic entities. The rules may comprise not only spatial information but also other kind of data (e.g., temporal) or their combination.
- ***Anchor-Point Proximity***: this notion is related to the concept of landmarks. There are several key locations, such as our home and work place, that we consider as *"anchor"* points in our understanding of the geographic environment where we live. In general, we may define an anchor-point as a frequently visited location or a location where one spends a lot of time.

In order to enhance recommender systems results, context may be used in different ways. In [2], the authors identify three forms of context-aware recommendation systems: Contextual pre-filtering (*PreF*), Contextual post-filtering (*PoF*) and Contextual modelling. In Section 4 we describe in more detail the first two.

3 Using the Web of Data to feed a Content-Based Recommender Systems

In the recent years, thanks to the `Web of Data` advance, we are witnessing a flourishing of semantic datasets freely available on the Web encoding machine-understandable `RDF` triples related to different domains and sometimes representing different points of view on the same domain. All this information can be exploited to model items and user profiles in an `LOD`-enabled content-based recommender system. One of the issues related to content-based approaches is the retrieval and pre-processing of the information used by the recommendation engine. In CB recommender systems, the module in charge of extracting relevant information from items description and representing it as a vector of keywords, is the so called *Content Analyzer* (CA) [15]. It usually uses some Natural Language Processing techniques to extract/disambiguate/expand keywords in order

to create a model of the item description. The use of LOD datasets to retrieve information related to an item eases the pre-processing steps performed by the CA since the information is already structured in an ontological way. Moreover, depending on the dataset, there is the availability of data related to diverse knowledge domains. If we consider datasets such as DBpedia or Freebase, we are able to access to rich linked data referring to a high variety of topics. Thanks to their SPARQL endpoints, we can quite easily extract portions related to the movie domain from LOD datasets. We use this information as the base for our content-based recommender system.

3.1 Computing Item Similarities in DBpedia

The main assumption behind our approach is that if two movies share some information (e.g., part of the cast, the director, the genre, some categories, etc.), then they are related with each other. Roughly speaking, the more features two movies have in common, the more they are similar. In a few words, a similarity between two movies (or two resources in general) can be detected if in the RDF graph: (1) they are directly related; (2) they are the subject of two RDF triples having the same property and the same object; (3) they are the object of two RDF triples having the same property and the same subject. Moreover, we exploit the ontological structure of the information conveyed within the Wikipedia categories, modeled in DBpedia by the properties dcterms:subject and skos:broader. This allows us to catch implicit relations and hidden information, i.e., information that is not directly detectable just looking at the nearest neighbors in the RDF graph. Fig. 1 shows a sample of the RDF graph containing properties and resources coming from DBpedia. In order to compute the similarities between movies, we adapted to an LOD-based setting one of the most popular models in classic information retrieval: the Vector Space Model (VSM). In this way we are able to represent items (i.e., movies) by means of feature vectors and then we can compute the similarity between them. The interested reader may refer to [11, 12] for further details.

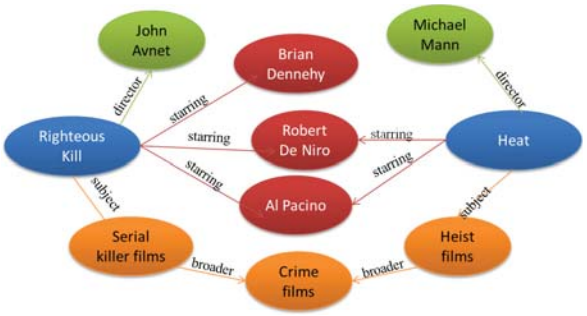


Fig. 1. A sample of an RDF graph related to the movie domain.

4 Cinemappy: a context-aware content-based RS

In this section we describe Cinemappy, a mobile application that implements a context-aware recommender engine. The purpose is to suggest movies and movie theaters to the users based on their profile and on their current location (both spatial and temporal). On the one side, the context-aware section of the system is implemented by adopting both a *PreF* and a *PoF* approach. In order to retrieve all the data needed to evaluate the geographical criteria presented in Section 2, the application leverages information from other freely available Web sources such as *Google Places*[1] or *Trovacinema*[2]. On the other side, the CB part of the recommendation engine exploits the computation of item similarities as described in Section 3. Moreover, driven by the context, the system also selects the right localized graph in DBpedia. Indeed, DBpedia contains also information extracted from localized versions of Wikipedia. Data coming from these Web sources are represented as different RDF graphs that can be easily selected via the FROM clause of a SPARQL query. The localized versions of DBpedia are particularly useful for the purpose of Cinemappy since some movies have for example a page in the Italian version of Wikipedia but they do not have a corresponding article in the English version. The basic building blocks of the system are depicted in Fig. 2. All the data about the descriptions of the movies are extracted via the

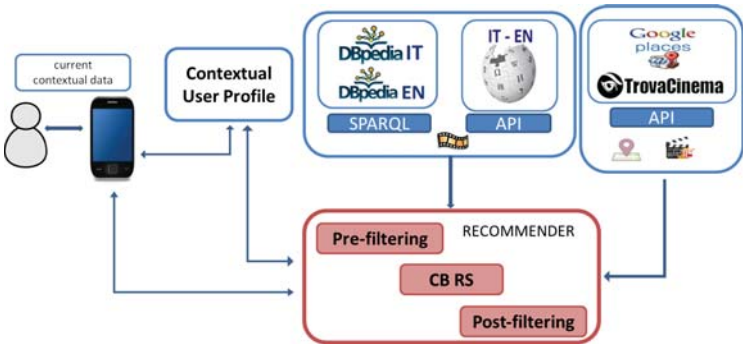


Fig. 2. System architecture.

DBpedia SPARQL endpoint. In particular, in our current implementation, we use both the *EN* and the *IT* graph since the application has been designed to be used by both English and Italian users.

Information about theaters and movies is extracted from *Trovacinema* while geographic data (latitude/longitude) about theaters has been obtained using the

[1] http://www.google.com/places/

[2] This is an Italian Web site where you can find information related to cinemas and scheduled movies – http://trovacinema.repubblica.it/.

Google Geocoding API[3]. In accordance with the *Co-location* principle, the system suggests POIs (Point of Interests) that are close to a given theater. In fact, the user might be interested in places close to the cinema such as restaurants, bars or playgrounds (if they are with their kids). The lists of POIs leverage the *Place Search* service that is part of the *Google Places API*[4]. These lists are created considering the relative distances between theaters and POIs and they use the *Google Distance Matrix API*[5]. In particular `Cinemappy` shows the POIs that are in a range of 2 km (about 1.24 miles) from a selected theater.

4.1 The recommender engine

`Cinemappy` uses several recommendation approaches to suggest movies and theaters to the user. Concerning contextual information we leverage both pre-filtering and post-filtering techniques for the contextual attributes introduced in Section 2. In particular to model the *Companion* attribute we use the so called *micro-profiling* approach [4], a particular pre-filtering technique. Basically, with *micro-profiling* we associate a different profile to the user depending on the selected companion. Also *Time* is used to pre-filter recommendation results. For geographical data we use a post-filtering approach. Between pre-filtering and post-filtering phases, to match movies with contextual user-profiles, we use the content-based recommendation strategy which leverages `DBpedia` as the only information source. Before we continue in the description of the system, a few words need to be spent on how we model the user profile. In our setting, the user profile is based on a binary rating such as *Like*/*Don't like* (as the one adopted by YouTube). Empirical studies on real uses cases, as the one reported in the official YouTube Blog[6], show that even if users are allowed to rate an item on a five stars scale, usually it happens that they either assign the highest score or do not give any feedback at all. Therefore, we model ratings using a binary scale.

Contextual Pre-filtering With `Cinemappy` we recommend bundles of items: *movies* to watch in *cinemas*. For this reason, movies that will not be featured in the future will not be suggested to the user. Nevertheless, such movies will be considered in the user profile if the user rated them. Moreover, for the current temporal and spatial position of the user, we constrain the set of movies to recommend considering geographical and time criteria. For each user u, the set of movies M_u is defined as containing the movies scheduled in the next d days in theaters in a range of k kilometers around the user position. The final recommendation list for u will be computed by considering only items available in M_u. This kind of restriction on the items with respect to time is a pre-filtering of the item set and not of the ratings as it usually happens in pre-filtering approaches.

[3] `https://developers.google.com/maps/documentation/geocoding/`
[4] `https://developers.google.com/places/documentation/`
[5] `https://developers.google.com/maps/documentation/distancematrix/`
[6] `http://youtube-global.blogspot.it/2009/09/five-stars-dominate-ratings.html`

Regarding the companion context, the micro-profiling approach is modeled by considering a specific profile for u for each companion cmp:

$$profile(u, cmp) = \{\langle m_j, v_j \rangle \mid v_j = 1 \text{ if } u \text{ likes } m_j \text{ with companion } cmp, \\ v_j = -1 \text{ otherwise}\}$$

In this way we are able to apply straightly the pre-filtering approach. When the user needs recommendations, given their current companion, the service considers only the corresponding micro-profile.

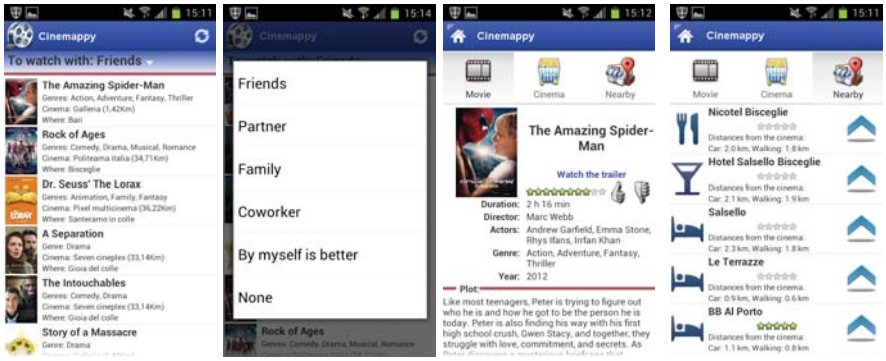
Content-Based Recommender The recommendation algorithm is based on the one proposed in [12], enhanced with micro-profiles management.

In order to evaluate if a movie $m_i \in M_u$ might be of interest for u given cmp we need to combine the similarity values related to each single property p of m_i and compute an overall similarity value $\tilde{r}_{PreF}(u_{cmp}, m_i)$:

$$\tilde{r}_{PreF}(u_{cmp}, m_i) = \frac{\sum_{m_j \in profile(u,cmp)} v_j \times \frac{\sum_p \alpha_p \times sim^p(m_j, m_i)}{P}}{|profile(u, cmp)|}$$

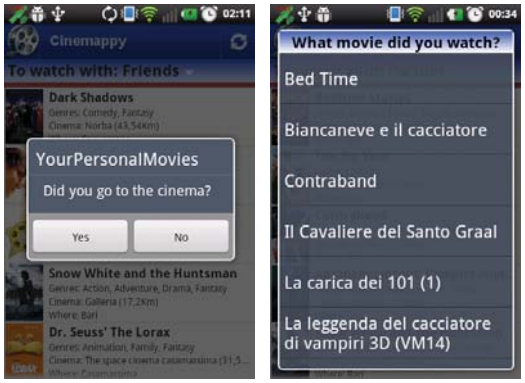
where P represents the number of properties in `DBpedia` we consider relevant for our domain (e.g. `dbpedia-owl:starring`, `dcterms:subject`, `skos:broader`, `dbpedia-owl:director`) and $|profile(u, cmp)|$ is the cardinality of the set $profile(u, cmp)$. The term $sim^p(m_j, m_i)$ represents the similarity between the two movies m_i and m_j with respect to a property p. The value is computed by adapting the Vector Space Model approach to an `RDF`-based setting. A weight α_p is assigned to each property representing its worth with respect to the user profile. The computation of these weights exploits machine learning techniques as described in [12]. Based on $\tilde{r}_{PreF}(u_{cmp}, m_i)$, we compute the ranked list $R_{u_{cmp}}$ of potential movies that will be suggested to the user.

Contextual Post-filtering Based on geographical criteria, we apply post-filtering on $R_{u_{cmp}}$ to re-rank its elements. In particular, for each criterion we introduce a $\{0, 1\}$-variable whose value is defined as follows:
`h` (*hierarchy*) : it is equal to 1 if the cinema is in the same city of the current user position, 0 otherwise;
`c` (*cluster*): it is equal to 1 if the cinema is part of a multiplex cinema, 0 otherwise;
`cl` (*co-location*): it is equal to 1 if the cinema is close to other POIs, 0 otherwise;
`ar` (*association-rule*): it is equal to 1 if the user knows the price of the ticket, 0 otherwise. This information is caught implicitly from the information about the cinema;
`ap` (*anchor-point proximity*): it is equal to 1 if the cinema is close to the user's house or the user's office, 0 otherwise.
These geographic criteria are combined together with $\tilde{r}_{PreF}(u_{cmp}, m_i)$ to obtain



(a) (b) (c) (d)

Fig. 3. Some screenshots of `Cinemappy`.



(a) (b)

Fig. 4. Location-Based Rating Acquisition Service.

a single score:

$$\tilde{r}(u_{cmp}, m_i) = \beta_1 \times \tilde{r}_{PreF}(u_{cmp}, m_i) + \beta_2 \times \frac{(\mathtt{h} + \mathtt{c} + \mathtt{cl} + \mathtt{ar} + \mathtt{ap})}{5}$$

where $\beta_1 + \beta_2 = 1$. In the current implementation of `Cinemappy`, both β_1 and β_2 have been chosen experimentally and have been set respectively to 0.7 and 0.3.

4.2 Implementation

`Cinemappy` has been implemented as a mobile application for Android smartphones[7]. When the user starts the application, `Cinemappy` displays a list of movies according to the current contextual user profile (Fig. 3(a)). The user can

[7] The `apk` installer is available at `http://sisinflab.poliba.it/cinemappy`.

choose their current companion from a list of different options thus enabling their micro-profile (Fig. 3(b)). Please note that even if the different user micro-profiles are empty, `Cinemappy` is able to suggest movies based exclusively on contextual information. For each movie in the list, its genres and the distance of the suggested theater from the user position are shown. Hence, the user can click on one of the suggested movies and look at its description, watch its trailer and express a preference in terms of *I would watch*/*I would not watch* (Fig. 3(c)). Furthermore, the user can find information about the recommended theater or the other theaters that feature that movie. Based on the theater location, the user could be interested in places where spending time with their friends, such as pubs, or with their girlfriend/boyfriend such as restaurants or bars, or with their family, and in this case maybe the user could be interested in certain kind of places also adequate for children. To support the user in this choice, the application suggests POIs by considering contextual criteria (Fig. 3(d)).

Location-Based Rating Acquisition Service User preferences acquisition plays a very important role for recommender systems in real scenarios. As previously pointed out, the system allows the user to rate movies while they are looking at their descriptions. Furthermore, thanks to the ubiquitous-awareness of mobile systems we are able to ask users to elicit their preferences in a more pervasive way. We exploit the geo-localization capabilities of mobile devices to understand if the user watched a movie. Every 90 minutes the application, by means of a background service, captures the user position. If the user has been for at least 90 minutes in a similar position close to a cinema in a time span corresponding to one or more scheduled movies, we assume that the user watched a movie in that cinema. In this case, the application asks the user if they went to the cinema (Fig. 4(a)) and and if a positive answer ensues, the user can rate one of the movies featured in that cinema (Fig. 4(b)).

5 Related Work

In this section we report on some of the main related works that we consider relevant to our approach and we classify them as *Context-Aware and Mobile Recommender Systems* and *Semantics in Recommender Systems*. A complete literature review in the two fields is out of the scope of this paper.

Context-Aware and Mobile Recommender Systems. As argued in [1], incorporating contextual information in traditional RSs is very important if we want to increase the quality of returned results. In the paper, the authors describe a multidimensional approach to recommendations, wherein the traditional user-item matrix is extended to a multidimensional model by adding new elements such as *place*, *time*, etc.. The context-aware recommendation process can take one of the three forms, depending on which stage of the process the context is applied in [2]. These forms are: *Contextual pre-filtering*, *Contextual post-filtering* and *Contextual modeling*. The *post-filtering* and *post-filtering* methods are compared in [20] where the authors, based on some experimental results, propose a simple but effective and practical way to decide how to use the two methods in a

recommendation engine. Context-aware recommender systems have attracted a lot of attention in mobile application. Mobile phones allow users to have access to a vast quantity of information in an ubiquitous way. In [22], the author details issues, scenarios and opportunities regarding mobile RSs especially in the area of travel and tourism. He describes the major techniques and specific computational models that have been proposed for mobile recommender systems. In [23] the authors describe *COMPASS*, a context-aware mobile tourist application where the context is modelled as the current user's requests. With regards to the profile, needs and context information of the user, *COMPASS* performs a selection of potentially interesting nearby buildings, buddies and other objects. These information change whenever the user moves or they change their goal. In [5] the authors present *ReRex*, a context-aware mobile recommender system that suggests POIs. By means of a Web-based survey application, the users are requested to imagine a contextual condition and then to evaluate a POI. In the answer the users have in mind the effect of the context on their decisions. The authors built a predictive model that is able to predict the relevance of such a POI in the contextual condition. Then, the application uses this model to allow the user to select the contextual factors and to browse the related context-aware recommendations.

Semantics in Recommender Systems. The need for a semantic representation of data and user profiles has been identified as one of the next challenges in the field of recommender systems [16]. Some ontology-based RSs are presented in [16] where the authors describe an approach that exploits the usage of ontologies to compute recommendations. Two systems, *Quickstep* and *Foxtrot*, are introduced that make use of semantic user profiles to compute collaborative recommendations. The profiles are represented by topics about research papers with respect to an ontology and the recommendations are computed matching the topics of the current user profile with the topics of similar users' profiles. The authors prove that: ontological inference improves the user profiling; the ontological knowledge facilitates the initial bootstrap of the system resolving the cold-start problem; the profile visualization improves the user profiling accuracy. A hybrid recommendation system is proposed in [8] wherein user preferences and item features are described by semantic concepts. These latter are clustered in order to obtain user's clusters corresponding to implicit Communities of Interest. In [18] the authors introduce the so called *semantically enhanced collaborative filtering* in which structured semantic knowledge about items is used in conjunction with user-item ratings to create a combined similarity measure for item comparisons. In [3] an approach is presented that integrates user rating vectors with an item ontology. In these works, the experiments prove an accuracy improvement over classical collaborative approaches also on the presence of sparse datasets. Most of the works described so far have been produced when `LOD` did not exist. The Web Of Data paves the way to the usage of new and rich semantic datasets to compute recommendations. In [13] the authors present a generic knowledge-based description framework built upon semantic networks. The aim of the framework is to integrate and to exploit some knowledge on several domains in order to compute cross-domain recommendations. They use a

spreading activation method with the purpose of finding semantic relatedness between items belonging to different domains. *dbrec* [21] is a music content-based recommender system leveraging the DBpedia dataset. They define the *Linked Data Semantic Distance* in order to find semantic distances between resources and then compute recommendations. In [11, 12] we present a model-based approach and a memory-based one to compute CB recommendations leveraging LOD datasets. Furthermore, we describe and compare several strategies to select ontological properties (in the movie domain) to be used during the computation of recommended items. A different hybrid technique is adopted in [17], where DBpedia is exploited as background knowledge for semantic tags recommendation. The semantic data coming from DBpedia are mixed with keyword-based information extracted via Web search engines to compute the semantic relatedness between the query posed by the user and the resources available in a semantic graph.

6 Conclusion and Future Work

In this work, we presented Cinemappy: a context-aware content-based recommender system for movies and movie theaters suggestions. The content-based part of the recommender engine is fed with data coming from localized DBpedia graphs and the results are enhanced by exploiting contextual information about the user. The application has been implemented as an Android application. Geographic criteria that go beyond the simple geographic distance have been implemented to fully exploit location-based information. Our future plans are: (a) evaluate the overall approach with real users; (b) enrich the information used by the content-based RS with other datasets in the LOD cloud; (c) apply the same approach to different context-aware domains such as tourist spots recommendations.

Acknowledgments. The authors acknowledge partial support of HP IRP 2011. Grant CW267313. The authors wish to thank Giosia Gentile for his contribution to development of Cinemappy.

References

1. G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, 2005.
2. G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253. 2011.
3. S. S. Anand, P. Kearney, and M. Shapcott. Generating semantically enriched user profiles for web personalization. *ACM Trans. Internet Technol.*, 7(4), Oct. 2007.
4. L. Baltrunas and X. Amatriain. Towards Time-Dependant Recommendation based on Implicit Feedback. In *Context-aware Recommender Systems Workshop at Recsys09*, 2009.
5. L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci. Context-aware places of interest recommendations for mobile users. In *HCI (9)*, pages 531–540, 2011.

6. L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci. Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing*, 16(5):507–526, 2012.
7. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semant.*, 7:154–165, September 2009.
8. I. Cantador, A. Bellogín, and P. Castells. A multilayer ontology-based hybrid recommendation model. *AI Commun.*, 21(2-3):203–210, 2008.
9. S. De Sabbata and T. Reichenbacher. Criteria of Geographic Relevance: An Experimental Study. *International Journal of Geographical Information Science*, 2012.
10. A. K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, Jan. 2001.
11. T. Di Noia, R. Mirizzi, V. C. Ostuni, and D. Romito. Exploiting the web of data in model-based recommender systems. In *6th ACM Conference on Recommender Systems (RecSys 2012)*. ACM, ACM Press, 2012.
12. T. Di Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker. Linked open data to support content-based recommender systems. In *8th International Conference on Semantic Systems (I-SEMANTICS 2012)*, ICP. ACM Press, 2012.
13. I. Fernández-Tobías, I. Cantador, M. Kaminskas, and F. Ricci. A generic semantic-based framework for cross-domain recommendation. In *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '11, pages 25–32, New York, NY, USA, 2011. ACM.
14. T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011.
15. P. Lops, M. Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. 2011.
16. S. E. Middleton, D. D. Roure, and N. R. Shadbolt. Ontology-based recommender systems. *Handbook on Ontologies*, 32(6):779–796, 2009.
17. R. Mirizzi, A. Ragone, T. Di Noia, and E. Di Sciascio. Ranking the linked data: the case of dbpedia. In *Proceedings of the 10th international conference on Web engineering*, ICWE'10, pages 337–354, 2010.
18. B. Mobasher, X. Jin, and Y. Zhou. Semantically enhanced collaborative filtering on the web. *Web Mining: From Web to Semantic Web*, pages 57–76, 2004.
19. C. Palmisano, A. Tuzhilin, and M. Gorgoglione. Using context to improve predictive modeling of customers in personalization applications. *IEEE Trans. Knowl. Data Eng.*, 20(11):1535–1549, 2008.
20. U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 265–268, New York, NY, USA, 2009. ACM.
21. A. Passant. Measuring semantic distance on linking data and using it for resources recommendations. In *Proceedings of the AAAI Spring Symposium "Linked Data Meets Artificial Intelligence"*, 3 2010.
22. F. Ricci. Mobile recommender systems. *J. of IT & Tourism*, 12(3):205–231, 2011.
23. M. van Setten, S. Pokraev, and J. Koolwaaij. Context-aware recommendations in the mobile tourist application compass. In P. D. Bra and W. Nejdl, editors, *AH*, volume 3137 of *Lecture Notes in Computer Science*, pages 235–244. Springer, 2004.

Ontology-based Rules for Recommender Systems

Jeremy Debattista, Simon Scerri, Ismael Rivera, and Siegfried Handschuh

Digital Enterprise Research Institute, National University of Ireland, Galway
firstname.lastname@deri.org

Abstract. Nowadays, smart devices perceive a large amount of information from device sensors, usage, and other sources which contribute to defining the user's context and situations. The main problem is that although the data is available, it is not processed to help the user deal with this information easily. Our approach is based on the assumption that, given that this information can be unified in a single personal data space, it can be used to discover and learn rules to provide the user with personal recommendations. In this paper we introduce a Rule Management Ontology to support the representation of event-based rules that trigger specific actions. We also discuss how a context listener component can provide recommendations based on the perceived context-data, or in the future, semi-automatically learnt rules.

1 Introduction

With the increasing popularity of smart devices, recommender systems can be useful in providing solutions based on real-time context information perceived by such devices in a ubiquitous environment. As McDonald pointed out, recommender systems "mediate the user experience in the digital world, and they will be increasingly helpful in performing the same role in the physical world, thereby filling an important gap in ubiquitous computing" [11]. Through sensors and device usage information, smart devices can be enabled to continuously accrue context information about repetitive daily tasks performed by the user; such as changing the mobile mode to silent when arriving at work, changing online status during a meeting, and opening certain applications and documents when in presence of other peers.

Data collected by smart devices is usually domain-specific to the application handling it. Having useful information in different domains and in different formats means that the heterogenous context data collected is not unified under one model. *Semantic heterogeneity* is thus another problem limiting the power of smart devices. Although different devices and their embedded sensors could be used for a common objective, their use of non-standard data formats means that the context information that they gather cannot be unified in an interoperable representation. This limitation, associated with the lack of a common domain model, is tackled in the di.me[1] project. The project targets the unification of the user's personal information across various heterogeneous sources, such as social networks and device sensors, into one personal data cloud. This is done with the aim of assisting the user in daily tasks. In the context of di.me, we

[1] http://www.dime-project.eu

restrict our working knowledge base to cover a personal closed-world environment; by extending and introducing models that enable the representation of the user's entire Personal Information Model (PIM). Unlike in Social Semantic Desktops [14], the PIM in di.me does not only cover conventional structured data (such as files, emails, status messages), but also unstructured and abstract data such as user context, situations, and online presence. This type of data is what will drive our intelligent recommender system to perform the desired functions through the di.me userware and intelligent user interface[2].

In this paper we will discuss our plans and progress in relation to the following three project objectives:

1. The definition of rules for recommendation and automation of tasks, based on the user's personal data cloud;
2. Automatic rule learning;
3. The processing mechanism to trigger learnt rules based on the perceived events.

For the first objective we provide a model that allows us to represent the learnt context-driven rules in a declarative manner. This contribution consists of the di.me Rule Management Ontology (DRMO)[3], an activity rule vocabulary integrated in the di.me knowledge representation models that allows PIM knowledge to be exploited for activity rule management. For the second objective, we are currently investigating techniques, such as case-based reasoning (CBR), which allow the system to automatically learn rules based on the user context-aware history. Currently we allow users to define context-aware rules using the intelligent user interface available in the di.me userware. The third objective is that of providing mechanisms to recommend actions to the user based on the real-time events perceived through the userware. We investigate various techniques in order to provide a scalable context listener which can provide recommendations or actions based on the learnt rules and the user's integrated PIM data.

After discussing the related work (Section 2), we discuss the integration of personal data in di.me and how open data sources such as LinkedGeoData[4] and Sindice[5] can be utilised for recommendation (Section 3.1). The Rule Management Ontology and the Context Listener are discussed in Section 3.2 and 3.3 respectively. We then discuss techniques for the automatic learning of context-driven rules (Section 3.4), before providing a real-world scenario (Section 4) and some concluding remarks and discussing future work (Section 5).

2 Related Work

In this section we look at how earlier efforts have tackled rule modelling, and how they were applied in various context-aware and event processing systems respectively. We also discuss techniques on how rules can be learnt automatically.

[2] The UI in question is currently being improved to factor in the results of a number of usability studies. A separate submission detailing its design is currently under review.

[3] http://www.semanticdesktop.org/ontologies/drmo/

[4] http://linkedgeodata.org/

[5] http://sindice.com/

In [4] and [2], rules are defined by an English-like rule language. The SECE system provided five different types of hard-coded events [4], and was later extended towards an ontology-based system [2] in order to enable the automatic discovery of services. This allowed the users to define rules that give recommendations based on open linked data services. The SECE language is specific to their system; thus, unlike our proposed model which is based on Resource Description Framework (RDF), the SECE rule language cannot be easily reused on other frameworks. The authors do not give any indication regarding the use of event and logical operators defined in the rule language. Li et al. [9] presents an event ontology with the aim of describing perceived situations. This model does not represent rules, but users can define rules by creating patterns (called processes) of composed event instances using event and logic operators. The resulting actions are inferred semantic knowledge, from the system knowledge base. On the other hand, in di.me, the actions give the user recommendations and automation based on the user's context-environment. In [10], May et al. present an ontology-based framework based on the Event-Condition-Action (ECA) pattern in order to integrate heterogenous semantic web services via rule definition. This framework allows the definition of sub-languages in order to process events from different web services, unlike the DRMO which is geared towards one domain: the personal information management. Sub-languages include detection processors for events defined in rules and for defining composite events. In contrast, in our model we create properties for composite conditions. May et al. defined the concepts Event-Condition-Action as three separate components since in a rule these might be defined with three different languages. In our case, the Event concept is composed by a number of condition blocks, and if these are satisfied they trigger one or more actions.

To apply our proposed rule model, we need a rule engine to execute the rules. In [9] event processes (rules) take perceived events as input and return an inferred *semantic event* as output. Their objective is to collect contextual data from the surroundings and present the user with an intelligent deduction of the current situation. An inference graph is used to keep track of the detected event concepts. In our context-listener we apply an *event-set* to store perceived events. We are also investigating time-window techniques with the aim of keeping the listener scalable. Rules are then executed on the *event-set* in order to provide recommendations. The framework proposed in [9] is composed of two inference engines; one which identifies rule patterns from perceived events and the other one which translates low-level event data into higher meaningful activities. In our case, the latter is being done by another part of the di.me framework, whilst for the former we do pattern matching using SPARQL[6] queries, since the event data we perceive in the userware is in RDF format. Using SPARQL for pattern matching allow instances of the DRMO ontology to execute on any triple store. Such technique was used by Teymourian and Paschke [16], where by using SPARQL to represent event patterns, they show how semantic events can be used in event detection. When an event is perceived, the system infers new triples using the system's event knowledge base. When the new inferred triples are created, they are sent to a rule-based inference engine and the engine decides if these triples should be stored in the system or discarded. In

[6] http://www.w3.org/TR/rdf-sparql-query/

di.me, SPARQL queries would not infer new knowledge, but indicate that rules should be triggered.

The proposed ontology (DRMO) can be used for automatic rule discovery in a context-aware environment. Similar approaches use ontologies as context models in Case-Base Reasoning (CBR) techniques [1, 8]. Bai et al. [1] proposed an ontology-based CBR (OntoCBR) to compare similarity between contexts, in order to reason and learn rules from a user's environment rather than rely on pre-defined or user-defined rules. In OntoCBR, user context is transformed into a situation case and is compared with other situations which are in the case-base using a semantic similarity measure. Thus recommendations are given if a situation has a match, in order to propose an action which has been carried out in similar situations. Knox et al. [8] make use of sensor data and CBR for activity recognition in a home environment. Cases are represented by an ontology which acts as a relationship between sensor data and resulting activities. The authors show that using CBR, various user activities can be learnt incrementally without having the need of training data. Similar to these two works, we intend to create a case-base from context data using the technique in [8], whilst learning new rules using similarity measures techniques as outlined in [1]. In order to ensure system scalability, we consider the work presented by Bergmann and Vollrath in [3], in which they discuss the idea of having a generalised CBR, where similar cases are generalised into one case in order to reduce the size of the case base.

3 Approach

In this section we first discuss how raw data from sensors and smart devices are integrated within the personal data cloud. We also discuss how linked open datasets such as LinkedGeoData and data source providers such as Sindice can be utilised to provide recommendations for the user, based on their current situation. We then introduce the di.me Rule Management ontology, before exploring how DRMO instances are mapped into SPARQL queries to simulate production rules. These rules are processed, and if their conditions are satisfied, the instructed actions are triggered to provide recommendations and automation in the di.me userware. Finally we discuss how we can learn and define DRMO rules automatically.

3.1 Exploiting Personal and Open Data Cloud for Intelligent Recommendation

The basic goal of the di.me userware is to learn about, gather and integrate the user's information and activities through their personal devices and online accounts, in order to provide a single-entry point to their personal data management and to provide context-aware recommendation and automation. di.me extends an interoperable knowledge representation format based on ontologies for the representation of personal information presented in [14], which also covers context-related personal knowledge. The use of this format allows interoperability with other applications that also utilise the RDF standard. Strang et al. recommends ontology-based modelling as the most appropriate way to engineer the core concepts in a context-aware environment [15]. After

having such personal information crawled and extracted from devices, it is semantically lifted onto the ontology-based PIM representation. The latter can then be used as a knowledge base for defining rules to provide recommendation and automation in the di.me userware.

Even though our primary focus is that of creating a big personal data cloud, external linked open data sources can be used to provide richer recommendations based on the user's current situation. LinkedGeoData could be utilised to, for example recommend a nearby restaurant for a user who is in a situation "Out of Office" during "lunch time". By gathering user context data surrounding the user, the system can invoke the LinkedGeoData SPARQL Endpoint to find restaurants located nearby to the current location of the user. When the data is returned, the userware might make use of a data source crawler such as Sindice to enrich the retrieved data by adding other information which might not be provided by the LinkedGeoData, such as restaurant ratings. Sindice is a web crawler which indexes statements in linked-data resources. This service helps users find certain resources and allows developers to integrate data from various datasources. LinkedGeoData and Sindice data sources can be combined with data in the personal cloud to provide better recommendation. The use of other datasets is also being investigated.

3.2 di.me Rule Management Ontology (DRMO)

The Rule Management Ontology (Figure 1) is inspired by approaches such as [10]. Rules are modelled on the Event-Condition-Action (ECA) pattern concepts. The ECA pattern is a structure used in event driven architectures, where the event part specifies on what event this rule might be triggered, the condition specifies under which conditions the actions should be triggered and the action part contains what is executed to lead the system to a new state, causing data to be changed [6]. Unlike the traditional architectures, DRMO rules are not specific to events (such as "on update" or "on delete"), but all rules can trigger actions if all of their conditions are satisfied. Thus, our pattern is defined as:

$$\textit{if } E[c_1, .., c_m] \implies [a_1, .., a_n] \tag{1}$$

where *event* E represents a rule that consists of a combination of *conditions* c, triggering one or more resulting *actions* a.

A rule is represented as a *drmo:Event*, which is composed (*drmo:isComposedOf*) of a number of *drmo:Condition* 'blocks' and triggers (*drmo:triggers*) one or more *drmo:Action* instances (Figure 1), similar to the work presented in [2, 4]. Thus, a DRMO event corresponds to an antecedent, whereas an action corresponds to the consequent part of a production rule.

In the ontology we define a number of different condition categories as subclasses of *drmo:Condition*. Since our focus is on the personal data cloud, the latter categories (*drmo:ResourceCreated, drmo:ResourceModified, drmo:ResourceDeleted*) represent the different changes affected in the Personal Information Model (PIM), such as receiving a new email (Resource Created), adding access control to peers on files (Resource Modified), and deleting a file (Resource Deleted). In the di.me userware we

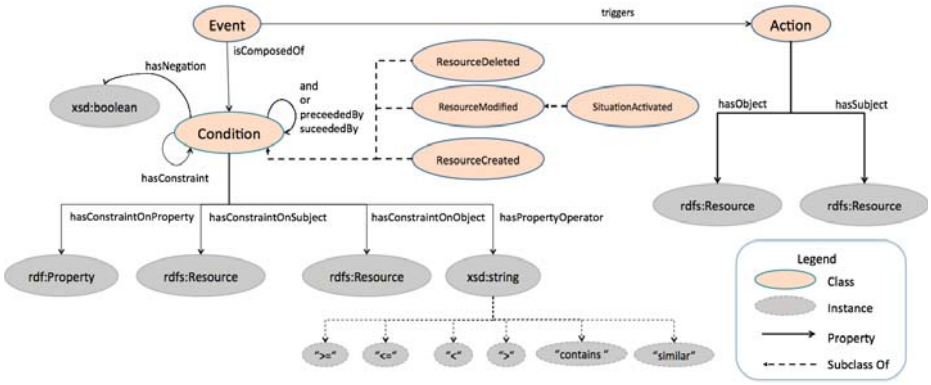


Fig. 1. The Rule Management Ontology

also allow for users to save personal activity context. A change in known user situations can be represented as an instance of *drmo:SituationActivated*, e.g. a change from situation "AtWork" to situation "AtHome". The different subclasses help the rule filtering mechanism according to the perceived system and user events.

Condition blocks can be composed of any information element (resource or class) on the Semantic Web. In di.me, condition blocks are restricted to elements from the OSCAF[7] ontologies and their attributes, and resources from the user's PIM. Multiple conditions are combined together using a number of logical (*drmo:and*, *drmo:or*) and event operators (*drmo:succeededBy*, *drmo:precededBy*). The set of operators are defined as properties in the DRMO (Figure 1) and are a subset of the operators used in [9] and defined in [5]. To explain these operators better, suppose conditions [C_1,C_2] occur at an instance of time t:

- Condition C_1 **AND** Condition C_2. Both conditions have to occur independent of **time** and **order** for the actions to trigger.

$$(C_1 \wedge C_2)(t) = (\exists t_1)(((C_1(t_1) \wedge C_2(t)) \vee (C_2(t_1) \wedge C_1(t))) \wedge t_1 \leq t) \quad (2)$$

- Condition C_1 **OR** Condition C_2. For the rule to trigger its actions, **only one** of the conditions have to be true at any given time.

$$(C_1 \vee C_2)(t) = (C_1(t) \vee C_2(t)) \quad (3)$$

- **SUCCEEDEDBY**(Condition C_1, Condition C_2). The actions are **only** triggered if Condition C_2 happens **at any time after** Condition C_1.

$$SUCCEEDEDBY(C_1, C_2)(t) = (\exists t_1)(C_1(t_1) \wedge C_2(t) \wedge (t_1 < t)) \quad (4)$$

- **PRECEDEDBY**(Condition C_2, Condition C_1). The actions are **only** triggered if Condition C_1 has happened **at any time before** Condition C_2.

$$PRECEDEDBY(C_2, C_1)(t) = (\exists t_1)(C_1(t_1) \wedge C_2(t) \wedge (t_1 < t)) \quad (5)$$

[7] http://www.oscaf.net

Each condition might also be negated using the *drmo:hasNegation* property. This property has a value range of either *true* or *false*. A condition may have one or more constraints (*drmo:hasConstraint*), for example *'if I receive an email from Anna'* the constraint here is that the rule will trigger only when an email from Anna is received. On the other hand, if a rule condition has a PIM resource such as a saved situation, then there might be no constraints, for example *'if I am AtWork'*. Generic conditions can also be defined without constraints, for example the rule *'if I receive an email'* would trigger always whenever a new email is received, irrelevant of the sender.

A condition can have three types of constraints: *drmo:hasConstraintOnProperty, drmo:hasConstraintOnSubject, drmo:hasConstraintOnObject*. These are used to define constraints on the event properties, for example the rule *'If I receive an email from anna@email.com'*, the value 'anna@email.com' is a constraint on the property *'nmo:messageFrom'*. To understand the DRMO constraint properties better we demonstrate the mentioned example as an instance in Listing 1.1:

```
... #prefix definitions

juan:event1 a drmo:Event ;
    drmo:isComposedOf juan:condition1 ; drmo:triggers juan:action1 .

juan:condition1 a drmo:Condition , nmo:Email ;
    drmo:hasConstraint juan:constraint1 .

juan:constraint1 a drmo:Condition ;
    drmo:hasConstraintOnProperty nmo:messageFrom;
    drmo:hasConstraintOnObject juan:constraint1:email1 .

juan:constraint1:email1 a   nco:EmailAddress , drmo:Condition ;
    drmo:hasConstraintOnProperty nco:emailAddress;
    drmo:hasConstraintOnObject  "anna@email.com" .

... #action definitions
```

Listing 1.1. An example of a user-defined rule

In Listing 1.1 the rule (*juan:event1*) is composed of *juan:condition1*, which is of type *drmo:Condition* and *nmo:Email*. *juan:contraint1* is added to the condition to represent the constraint given by the rule *'if I receive an email **from anna@email.com**'*. This is described in the rule instance as a *drmo:hasConstraintOnProperty* 'nmo:messageFrom' and as a *drmo:hasConstraintOnObject* 'anna@email.com'. The URI 'juan:constraint1:email1' is not a resource from the PIM, but it is a generated resource of type *drmo:Condition* since nmo:messageFrom would be expecting a URI of a resource (e.g. 'mailto:anna@email.com') rather than the value itself. These properties help to transform rules from a DRMO instance into SPARQL queries, as we discuss in Section 3.3. The *drmo:hasConstraintOnSubject* property allows for implicit values such as resources from the PIM, for example 'PIM:Anna' to be used in rule constraints. This allow the rule to be triggered every time an email from Anna is received, if her email address is in the user's PIM.

A constraint might also have relational operators in order to compare the perceived event values with the value of the rule's condition (e.g. *"if I receive an email and the subject contains di.me"*). In di.me, "contains" and "similar" can be used as string operators,whereas we define operator instances for numeric datatypes such as $\leq$, $\geq$, $<$, and $>$.

The *drmo:Action* class specifies an action instance (e.g. *Recommend*), whose semantics are understood by the system and result in specific actions. The *drmo:hasSubject* specifies the receiver of the action whilst the *drmo:hasObject* specifies what parameters need to be passed to the executed actions. The parametrisation of action subjects and objects is similar to that defined in [13], where action instances are used to classify emails. Examples of action instances could be recommending a nearby restaurant, where the action instance 'Recommend' would have "Restaurant" as a subject and *pimo:Location* as an object.

3.3 A Context Listener for Rule Activation

A context listener is required to register defined rules and to activate them when their conditions are satisfied by the perceived events. Rule instances are transformed into SPARQL queries and registered to a rule pool in the context-listener. In this section we will first explain how rules are transformed from DRMO instances into SPARQL. Then we show how event processing in the context-listener is done.

Transforming rules from DRMO instances to SPARQL - Rules are defined as instances of the DRMO. As part of the event processing mechanism, the condition part of the rule is transformed into SPARQL queries which might be executed at a specific time window, or each time an event is registered in the PIM. Since we also have logic operators, we discuss how these will be parsed using an infix to postfix technique. For each rule instance in the user's PIM, the transformer maps the *drmo:Condition* blocks to SPARQL queries. The properties *drmo:hasConstraintOnSubject, drmo:hasConstraintOnProperty* and *drmo:hasConstraintOnObject* are mapped to a SPARQL triple pattern {?subject ?predicate ?object . } respectively. A negated condition block (*drmo:hasNegation*) is mapped into the SPARQL query combining[8] 'OPTIONAL', 'FILTER' and 'BOUND'. The negation function is still a feature recommendation for SPARQL 1.1[9]. The transformer parse logical operators *drmo:and* and *drmo:or* in precedence ordering. We decided to implement a postfix technique since it allows us to define the order of execution from an infix notation. Multiple conditions composed with the *drmo:or* property have the condition triples separated with the 'UNION' keyword, whilst those with the *drmo:and* have their condition triples joined in one query. SPARQL 'FILTER's are used when a constraint has the *drmo:hasPropertyOperator* defined. Filters are also used when multiple conditions are composed together using *succeededBy* and *precededBy* (Section 3.2), by restricting patterns using timestamps. The action instances are stored in a separate object together with the rule instance URI, and not part of the SPARQL query. Listing 1.2 shows how the rule instance in Listing 1.1 is transformed into a SPARQL query.

```
SELECT * WHERE {
    ?_cn61 a nmo:Email .
    ?_cn61 nmo:messageFrom ?_varFE208 .
    ?_varFE208 a nco:EmailAddress .
```

[8] http://www.w3.org/TR/rdf-sparql-query/#func-bound
[9] http://www.w3.org/TR/sparql-features/

```
    ?_varFE208 nco:emailAddress 'anna@email.com' .
}
```

Listing 1.2. SPARQL Transformation for rule in Listing 1.1

Event Processing and Pattern Matching - When a rule is transformed to a SPARQL pattern, it is then registered to a rule pool in the context listener (Figure 2). The transformation of DRMO instances into SPARQL queries help us in the pattern matching process. Triple patterns can be considered as the *antecedents* of a production rule, since instance conditions are transformed into triple patterns that have to be matched in the *working memory*. Our working memory consists of an *event-set* which holds events perceived by the userware and the PIM. Each time an event is perceived, the event data is timestamped and stored in the *event-set*. Using the new perceived information,

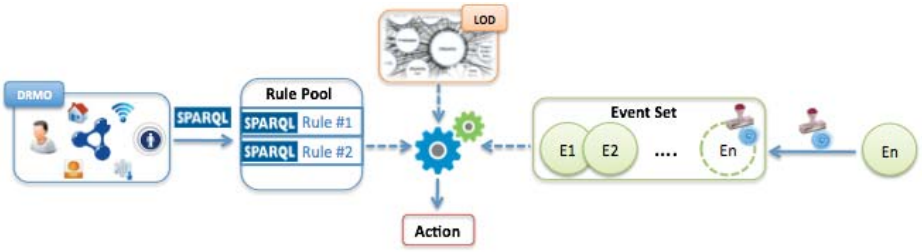


Fig. 2. Context Listener

the context listener will then filter rules by category (see Section 3.2) and type (for example 'nmo:Email') to keep only the ones that might be triggered from the perceived events stored in the set. From these filtered rules, SPARQL queries are executed and results are returned, triggering satisfied rules and thus performing the specified actions. Since this is a real-time listener and events are continuously perceived, we need to cater for events which happened in the past and are no longer required in the Event-Set. The use of a *time window* helps us keep the listener scalable by removing past perceived events, for example, keeping in memory only events that happened in the last hour. An evaluation will be carried out to find the most appropriate time-window limit for our usecases. To improve the scalability and efficiency of our context-listener, we are currently investigating how algorithms such as Rete [7], which speeds up the matching process, can be integrated in this module for optimisation purposes.

3.4 Learning Rules from Context Data

As discussed in previous sections, the semantic framework in the di.me userware gathers context-related information about the user. Perceived context data is semantically lifted onto a unique 'live context' instance of the Context Ontology (DCON)[10] [12]. In

[10] http://www.semanticdesktop.org/ontologies/dcon/

addition, past context snapshots can be timestamped and made persistent as instances of the User History Ontology (DUHO)[11]. Thus it is possible to construct a timeline of events and corresponding actions, as part of a user's history. Using CBR techniques mentioned in the related work section, these instances will be analysed to find similar situation-action patterns, in order to learn and define rule instances automatically. As a simple example, user John frequently forwards emails with a subject containing "ISWC" to his student Mary. By analysing the context logs, the system could identify this pattern and learn this rule. From then onwards, whenever John receives an email with the subject containing "ISWC", the system will recommend John to automatically forward it to Mary.

4 A Rule Scenario

In this section we provide a practical scenario of how a rule instance is defined in the PIM. We show how our ontologies can represent rules for context-aware recommendation. Listing 1.3 provides an example of a rule, represented as an instance of DRMO. The rule represents the scenario *If I find myself in situation "Out of Office", and it is "Lunch Time", recommend me a nearby restaurant*. In this rule we see that juan:event34, which is an existing resource in the PIM, is composed of two conditions, juan:condition8 and juan:condition9. The former refers to the previously saved situation "Out of Office" (<urn:juan:graph:situation14>), and the latter refers to a more complex condition. Here, the *LiveContext* is queried to get the current time. Both condition items are existing resources in the PIM. If the current time instance points at Lunch Time (defined in the PIM), then the action is triggered. "Recommend" is a system defined action that is used by Juan. The action has two properties, juan:subject1 and juan:object1. The former defines the type of recommendation needed and the latter passes any parameters to satisfy the recommendation. The system will transform the instance and parameters into a SPARQL query to an open data endpoint such as the LinkedGeoData which returns a list of possible restaurants near the user's current location.

```
@prefix drmo: <http://www.semanticdesktop.org/ontologies/2012/03/06/drmo#> .
@prefix pimo: <http://www.semanticdesktop.org/ontologies/2007/11/01/pimo#> .

juan:event34 a drmo:Event ;
  drmo:isComposedOf juan:condition8, juan:condition9 ; drmo:triggers juan:action4
       .

juan:condition8 a drmo:Condition ;
   drmo:hasConstraint <urn:juan:graph:situation14> ;
   drmo:and juan:condition9 .

juan:condition9 a drmo:Condition , dcon:LiveContext ;
  drmo:hasConstraint juan:constraint12 ;
  drmo:and juan:condition8 .

juan:constraint12 a drmo:Condition ;
  drmo:hasConstraintOnProperty dcon:currentTime ; drmo:hasConstraintOnObject <urn:
       juan:PIM:LunchHours> .

juan:recommend a drmo:Action ;
    drmo:hasSubject juan:subject1 ;
```

[11] http://www.semanticdesktop.org/ontologies/duho/

```
    drmo:hasObject juan:object1 .

juan:subject1 a rdfs:Literal ;
    rdf:value "Restaurants"^^xsd:string .

juan:object1 a pimo:Location ;
    pimo:hasLocation <urn:juan:PIM:CurrentLocation1> .
```

Listing 1.3. An example of a user-defined rule

5 Future Work and Concluding Remarks

In this paper we described our approach for an ontology-driven recommender system that suggests actions to the user, driven by context and knowledge from their aggregated personal data cloud. We described how personal data from various sources can be combined with the user's sensed situational context in order to detect recurring situations and a vast range of associated actions that can be fully or partly automated. We also explained how open link data services like Sindice, and data sources such as LinkedGeoData, can also be exploited by the system to recommend items that are not part of the user's personal data cloud, i.e. suggest new, possibly unknown items, to the user. A rule management ontology, integrated with existing standard ontologies for the comprehensive modelling of distributed personal information, is proposed. The ontology supports the definition of context-driven recommendation rules that are learnt by an intelligent system, or defined semi-automatically by the user. The rules are converted from RDF to SPARQL queries at runtime by a context listener, which continously matches them against perceived user activities and system events. A matched rule results in a recommendation being provided to the user.

Currently the implemented protoype only caters for user-defined rules, that can be intuitively constructed through an intelligent UI. In the future, techniques to enable the automatic discovery of rules will be further investigated. Further enhancements under consideration are techniques such as the Rete algorithm for a more efficient context listener (performance-wise), and the investigation and evaluation of techniques for managing rule conflicts. To determine the effectiveness and usability of the proposed system, we will perform evaluation to determine i) how many different kinds of use-cases the DRMO ontology is able to cover, ii) the ideal time-window to ensure the context-listener's scalability and efficiency, iii) the user's own assessment of the system's usability and the adequacy of the resulting in-context recommendations.

Acknowledgments. This work is supported in part by the European Commission under the Seventh Framework Program FP7/2007-2013 (*digital.me* – ICT-257787) and in part by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (*Líon-2*).

References

1. Y. Bai, J. Yang, and Y. Qiu. Ontocbr: Ontology-based cbr in context-aware applications. In *Proceedings of the 2008 International Conference on Multimedia and Ubiquitous Engineering*, MUE '08, pages 164–169, Washington, DC, USA, 2008. IEEE Computer Society.

2. V. Beltran, K. Arabshian, and H. Schulzrinne. Ontology-based user-defined rules and context-aware service composition system. In *Proceedings of the 8th international conference on The Semantic Web*, ESWC'11, pages 139–155, Berlin, Heidelberg, 2012. Springer-Verlag.
3. R. Bergmann and I. Vollrath. Generalized cases: Representation and steps towards efficient similarity assessment. In *Proceedings of the 23rd Annual German Conference on Artificial Intelligence: Advances in Artificial Intelligence*, KI '99, pages 195–206, London, UK, UK, 1999. Springer-Verlag.
4. O. Boyaci, V. Beltran, and H. Schulzrinne. Bridging communications and the physical world: sense everything, control everything. In *Proceedings of the 5th International Conference on Principles, Systems and Applications of IP Telecommunications*, IPTcomm '11, pages 14:1–14:6, New York, NY, USA, 2011. ACM.
5. S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S.-K. Kim. Composite events for active databases: Semantics, contexts and detection. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 606–617, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.
6. K. R. Dittrich, S. Gatziu, and A. Geppert. The active database management system manifesto: A rulebase of adbms features. In *Proceedings of the Second International Workshop on Rules in Database Systems*, RIDS '95, pages 3–20, London, UK, UK, 1995. Springer-Verlag.
7. C. L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17 – 37, 1982.
8. S. Knox, L. Coyle, and S. Dobson. Using ontologies in case-based activity recognition. In *FLAIRS Conference*, 2010.
9. Z. Li, C.-H. Chu, W. Yao, and R. A. Behr. Ontology-driven event detection and indexing in smart spaces. In *Proceedings of the 2010 IEEE Fourth International Conference on Semantic Computing*, ICSC '10, pages 285–292, Washington, DC, USA, 2010. IEEE Computer Society.
10. W. May, J. J. Alferes, and R. Amador. An ontology- and resources-based approach to evolution and reactivity in the semantic web. In *Proceedings of the 2005 OTM Confederated international conference on On the Move to Meaningful Internet Systems: CoopIS, COA, and ODBASE - Volume Part II*, OTM'05, pages 1553–1570, Berlin, Heidelberg, 2005. Springer-Verlag.
11. D. McDonald. Ubiquitous recommendation systems. *Computer*, 36(10):111 – 112, oct. 2003.
12. S. Scerri, J. Attard, I. Rivera, M. Valla, and S. Handschuh. Dcon: Interoperable context representation for pervasive environments. In *In Proceedings of the Activity Context Representation Workshop at AAAI 2012*, 2012.
13. S. Scerri, G. Gossen, B. Davis, and S. Handschuh. Classifying action items for semantic email. In *LREC*, 2010.
14. M. Sintek, S. Handschuh, S. Scerri, and L. van Elst. Technologies for the social semantic desktop. In *Reasoning Web. Semantic Technologies for Information Systems*, volume 5689 of *Lecture Notes in Computer Science*, pages 222–254. Springer Berlin / Heidelberg, 2009.
15. T. Strang and C. L. Popien. A context modeling survey. In *UbiComp 1st International Workshop on Advanced Context Modelling, Reasoning and Management*, pages 31–41, Nottingham, September 2004.
16. K. Teymourian and A. Paschke. Semantic rule-based complex event processing. In *Proceedings of the 2009 International Symposium on Rule Interchange and Applications*, RuleML '09, pages 82–92, Berlin, Heidelberg, 2009. Springer-Verlag.

Ontology-centric decision support

Marco Rospocher and Luciano Serafini
FBK-irst, Via Sommarive 18 Povo, I-38123,Trento, Italy

Abstract. In the last few years, ontologies have been successfully exploited by Decision Support Systems (DSSs) to support some phases of the decision-making process. In this paper, we propose to employ an ontological representation for *all* the content both processed and produced by a DSS in answering requests. This semantic representation supports the DSS in the whole decision-making process, and it is capable of encoding (i) the request, (ii) the data relevant for it, and (iii) the conclusions/suggestions/decisions produced by the DSS. The proposed approach offers many advantages, and have been successfully implemented in an DSS for personalized environmental information, developed in the context of the PESCaDO EU project.

1 Introduction

Decision support systems (DSSs) are information systems that support users and organizations in *decision-making* activities. DSSs have been applied in several diverse application contexts, to help to take decisions in domains like the medical, legal, computer security, and power consumption management ones.

At an abstract level, we can identify three phases in a decision-making process [1]:

1. the formulation of the decision-making problem;
2. the gathering, storing, and fusion of the data relevant for the given problem;
3. the reasoning on the data to take a decision;

To support the implementation of such process, DSSs usually comprise three main modules [2]: the (i) *dialogue* or *user* module, which supports the interaction of the user with the system, to formulate the problem and receive in output the result of the the DSS computation, the (ii) *data* module, which allows to store the data collected and processed by the DSS, and the (iii) *model* module, which implements the decision support strategy.

Being studied, both theoretically and technically, since the late 1960s, research in DSSs field has taken advantage in the last decade of the achievements and results of Semantic Web technologies. In particular, ontologies have recently been adopted in DSSs in various application domains [3,4,5,6,7,8,9,10], exploited for several purposes: to support via reasoning some of the decision support phases [6,3,8], to characterize the data manipulated by the DSS [11,3], and to define the tasks and parameters of the various modules of the system [9]. That is, so far ontologies have been adopted by DSSs to support only parts of the decision-making process, mainly to represent the data and to support their processing for taking decisions.

In this paper we propose to exploit an ontology-based knowledge base as the main (enhanced) data structure of a DSS, where *all* the content and data for a specific decision support request, processed and produced by the system, are stored. In details, our approach consists in designing the ontology underlying the knowledge base, i.e. the *T-Box* in Description Logics (DL) terminology, so that it is capable of formally representing all the details of the three decision-maikng phases described above, i.e., (i) the decision support request submitted by the user to the system, (ii) the data that the system processes for the given request, and (iii) the new content and conclusions produced by the DSS from the available data and in view of the given request, possibly together with some details on how the DSS arrived to those conclusions.

Each single request submitted to the system triggers the instantiation of a new *A-Box* of this ontology. The instantiation incrementally occurs in subsequent steps, coherently with the decision-making process phases. Therefore, at the end of the processing of a request by the DSS, the A-Box associated with the request contains a structured and comprehensive description, a *semantic request story-plan*, of the output produced by the DSS, linked to the data and the request that triggered that output.

The ontological representation of the DSS data that we propose is used to support the DSS activities

- as the main, shared, data structure of the DSS;
- as content exchange format between the different modules of the DSS;
- to keep track of all the intermediate data and results produced by the DSS in the course of solving a problem.

The advantages of using a semantic (ontology based) representation of the main data structure of a DSS are many. First, differently from what happened in the past where DSS where closed systems, in the semantic web era most of the knowledge and data useful to support a decision is available (in heterogeneous formats) in the web. As one of the main objectives of ontologies is to define shared domain models, an ontology-based representation of the knowledge in a DSS facilitates the integration of structured knowledge and data available in the web. Second, in the semantic services era we are now, a DSS can be seen as any other web-service and therefore it can be combined with other semantic services. Keeping a semantically rich track of the entire decision process followed by a DSS in order to reach a conclusion/suggestion/decision, and exposing it by adopting for instance the Linking Open Data[1] principles, enables the combination of the DSS with other complex services, such as explanation services (for which, just information about input-output is not enough) or case reuse/adaptation services (which can adapt the entire reasoning chain done by the DSS to slightly different cases). Finally, the third advantages is the fact that some of the inference steps of the DSS can be performed via state of the art logical reasoning services, as for instance rule engines or ontology reasoners.

The proposed approach has been successfully applied in a running personalized environmental DSS, in the context of the PESCaDO EU project, where its features and advantages have been empirically demonstrated.

[1] http://linkeddata.org/

2 The PESCaDO Use Case

Citizens are increasingly aware of the influence of environmental and meteorological conditions on the quality of their life. One of the consequences of this awareness is the demand for high quality environmental information and decision support that is tailored (i.e., personalized) to one's specific context and background (e.g., health conditions, travel preferences). Personalized environmental information may need to cover a variety of aspects (e.g., meteorology, air quality, pollen) and take into account a number of specific personal attributes of the user (e.g., health, age, allergies), as well as the intended use of the information.

The goal of the PESCaDO EU project[2] is to develop a multilingual web-service platform providing personalized environmental information and decision support. This platform takes advantage of the fact that nowadays, the Web already hosts a great range of *environmental nodes* (i.e. web-sites, web-services, open data repositories) that offer data on each of the above aspects, such that, in principle, the required basic data are available. The challenge is thus threefold: first, to discover and orchestrate these environmental nodes; second, to process the obtained data in accordance with the decision support needs of the user; and, third, to communicate the gained information in the user's preferred mode.

For a general overview of the running PESCaDO system[3], and the type of information produced, check the demonstration video[4], or directly play with the on-line demonstrator[5]. Shortly, users submit a decision support request to the system (e.g. "*I want to do some hiking in Nuuksio Park tomorrow: is there any health issues for me?*"), specifying in full details the type of request, the type of activity (if any) they want to perform, their profile, the geographic area and the time period to be covered. Then, the system (i) determines the data relevant for the request, (ii) retrieves the data from environmental nodes providing them[6], (iii) processes these data providing conclusions (e.g., warnings, recommendations) according to the needs of the users, and, finally, (iv) generates reports (e.g., text, tables, graphics) to be communicated to the user.

3 The Decision Support Knowledge Base

We propose a reference architecture for designing ontology-based knowledge bases for decision support systems, called Decision Support Knowledge Base (DSKB). It aims at guiding the development of an OWL [13] ontology capable of representing in a connected and comprehensive way all the content relevant for a given decision support request. In details, in our approach each decision support request is associated with an A-Box (i.e., a set of individuals and assertions on them) instantiating the T-Box part of the DSKB (see Figure 1[7]). The DSKB T-Box, to which we refer to as *Deci-*

[2] http://www.pescado-project.eu

[3] A more comprehensive description of the system workflow can be found in [12]

[4] http://www.youtube.com/watch?v=c1Ym7ys3HCg

[5] Accessible from the project web-site, or directly here http://193.145.50.130/

[6] More precisely, data are hourly distilled from web-sites and stored in a dedicated repository, so when a decision request is submitted to the system, this database is actually queried for data.

[7] The three parts of each request A-Box correspond to the components of the T-Box.

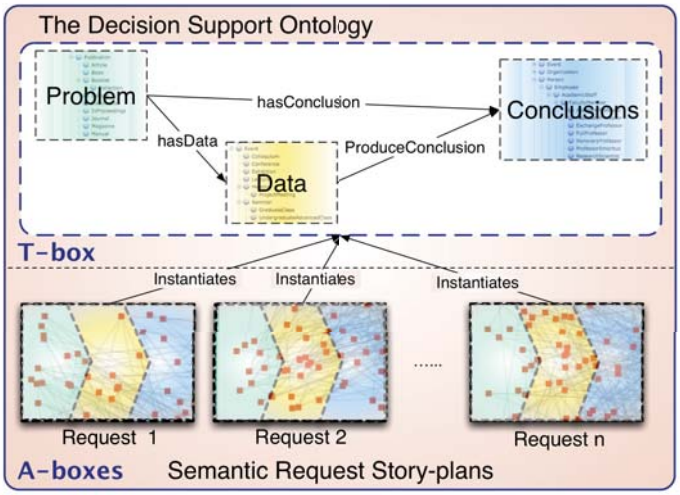


Fig. 1: The Decision Support Knowledge Base

sion Support Ontology (DSO), comprises three main components, namely Problem, Data, and Conclusions. These three components are connected by relations between the corresponding elements. As shown by Figure 1, these relations are[8]: *hasData* and *hasConclusion*, which relates a problem description with the data relevant for it and the conclusions provided for it by the DSS, and *ProduceConclusion* which connects the data with the conclusions they trigger. As we will remark in Section 3.4, these object properties allow to relates the instances of the different components of the DSO, to assembly a connected semantic request story-plan. Furthermore, these properties are particularly useful for explanation purposes: e.g., the *ProduceConclusion* property allows to keep track of what data triggered a certain conclusion of the DSS.

Next, we describe each part of the DSKB, reporting the details of its implementation in the context of the PESCaDO DSS[9].

3.1 Problem component

The purpose of this component of the DSKB is to formally describe all the aspects of decision support problems that the user can submit to the system. In its simplest form, this component could consist in a taxonomy of the request types supported by the system, enriched by the additional input parameters that are needed by the DSS to provide adequate decision support to the users (e.g., date/time of the request). In more advanced situations, a problem description may include also aspects of the users profile

[8] The set of object properties here presented is not exhaustive, and can be further extended depending also on the application context.

[9] The DSO of the PESCaDO DSKB consists of 210 classes, 99 object properties, 42 datatype properties, and 641 individuals, and it is available at: `http://www.pescado-project.eu/ontology.php`

(e.g., age, preferences, diseases), or other additional problem features that may affect the decision support provided by the DSS. This component may also be used by the DSS dialogue module to guide (and constrain) users in composing a request.

PESCaDO Problem component The Problem component defined for the PESCaDO system comprises three interrelated parts: *Request*, *Activity*, and *User*. *Request* describes a taxonomy of request types supported by the system (e.g., "Is there any health issue for me?", "Do environmental conditions require to take some administrative actions?"). *User* enables to describe the profile of the user involved in the request. Examples of the aspects modelled in this component are the user typology (e.g., end-user or administrative user), the age of the user, the gender, diseases or allergies the user may suffer from. Finally, *Activity* describes the activities the user may want to undertake, and that may affect the decision support provided by the system. For instance, different factors are considered by the DSS if the user decides to do some physical outdoor activity rather than travelling with public transportation.

These three sub-components are interrelated by object properties (e.g., a request has to have a user profile associated with it, and may involve an activity the user wants to undertake) and axioms that constrain the possible combinations allowed. The PESCaDO user interface module has been developed to dynamically read this constraints from the ontology, to guide the users in formulating their decision support problems. For instance, the subclass restriction "hasRequestUser only AdministrativeUser" on request class "CheckAirQualityLimits" states that the request for checking the air quality limits can be submitted only by administrative users. Similarly, the restriction "hasRequestActivity some (AttendingOpenAirEvent or PhysicalOutdoorActivity or Travelling)" on request class "AnyHealthIssue" enables to propose to the users only some activities, those of type "AttendingOpenAirEvent" or "PhysicalOutdoorActivity" or "Travelling", forcing them to select one of those. Further parameters are also defined to allow the specification of all the necessary details to compose a complete problem description: for instance, the time period and geographical region considered in the request.

3.2 Data component

The purpose of this component of the DSKB is to formally describe the data accessed and manipulated by the DSS. For instance, in the case of an environmental DSS, the Data component could describe physical phenomena observations like temperature, humidity, or wind speed, while in the case of a financial DSS, it may represents stock market rates or currency exchanges.

To some extent, this component play the role of the *domain ontology* of the DSS application. Differently from the other two components of the DSO which are more application-oriented, an ontology to be used as Data component may be already available in the web, and thus reused in the DSKB. By adopting an ontological representation of the data processed by the DSS, we favour the integration of (structured) data provided by heterogeneous sources, like web-sites or nodes of the Linking Open Data cloud. For instance, this approach enables to easily exploit for decision making purposes the data exposed by smart city initiatives, like the SmartSantander project[10].

[10] http://www.smartsantander.eu/

All the aspects of the data that may affect the conclusions taken by the DSS should be described in this component: e.g., validity, provenance, trust, uncertainty. For instance, the Data component may incorporate standardization efforts like the Open Provenance Model [14].

PESCaDO Data component The Data component in PESCaDO describes the environmental data used by the system to provide decision support: e.g., meteorological data (e.g., temperature, wind speed), pollen data, and air quality data (e.g., NO2, PM10, air quality index). All the necessary details to comprehensively represent observed, forecast, or historical data are included, such as values (both quantitative and qualitative values are supported), the time period covered by the data, and the type of the data (e.g., instantaneous, average, minimum, maximum). Concerning the values, the mapping between qualitative values and quantitative ones is also encoded in the ontology: for instance, a *moderate* quantity of birch pollen in the air correspond to a concentration between 10 and 100 grains per meter cube of air. Detailed information on the environmental node providing the data is also representable, like its type (e.g., measurement station, web-site, web-service), geographical location, and confidence value.

The development of the Data component in PESCaDO has involved (i) the reuse of (part of) some already available environmental domain ontologies (e.g., SWEET[11]), (ii) the application of techniques for automatic ontology extension [15] (e.g., to define the pollen sub-domain), and (iii) the contribution of environmental domain experts.

The ontological representation of the data processed by the PESCaDO DSS is exploited by a component of the system (the Data Fusion Module) to integrate, for decision-making purposes, the input data coming from heterogeneous sources and obtained with different techniques, like by querying environmental web-services or by distilling data from text and images of environmental web-sites.

3.3 Conclusions component

The purpose of this component of the DSKB is to formally describe the output produced by the DSS by processing the problem description and the data available. Examples of the content to be produced are warnings/suggestions/instructions, as well as the results of further processing of the data (e.g., data aggregations, data analysis results). Details on the confidence of the system about this content may also be represented, for instance supporting the possibility to assign a weight. Furthermore, if needed, this component may also allow to represent the feedback left by the users about their degree of satisfaction of the content produced by the DSS for the submitted request. We also recall that the "ProduceConclusion" property links the conclusions produced by the DSS to the data that triggered them, an information that may be exploited for explanation purposes.

PESCaDO Conclusions component The Conclusions component in PESCaDO allows to describe conclusions like warnings, recommendations, and suggestions that may be triggered by environmental conditions, or exceedances of air pollutants limit values that may be detected from the data. An example of warning type encoded in

[11] http://sweet.jpl.nasa.gov/ontology/

the PESCaDO DSKB is the one to be triggered if the hourly NO2 concentration exceeds the EU limit value, having the following message text attached: "*Nitrogen dioxide causes respiratory symptoms especially in children and asthmatics, because high concentrations of this gas cause contraction of the bronchial airways. It may increase the sensitivity of the airways to other irritants such as cold air and pollen.*"

A warning issued by the PESCaDO system for a given decision support request is represented as a new instance in the A-Box associated with the request, having an object property asserting the type of warning, and a datatype property asserting the relevance (a decimal value in [0..1]) of the warning for the current problem request.

3.4 Semantic request story-plans

The three components of the DSO provide a schema to represent the main aspects of a decision support request. Therefore, for any given decision support request submitted to the system, the actual content about these three aspects of the given request can be formalized as a set of individuals, and assertions on them, instantiating the T-Box part of the DSKB. This results in a connected set of triples, what we called a semantic request story-plan. A semantic request story-plan is an RDF graph covering all the aspects of any decision support request: its formulation, the data relevant for it, and the conclusions generated by the DSS from the data.

An A-Box of the PESCaDO DSKB Figure 2 shows an excerpt of an A-Box instantiating the proposed schema. The three blocks in Figure 2 corresponds the three components of the T-Box, of which the subjects of the assertions instantiate some classes. Note the connections between the individuals of different components, highlighted in (red) boxes and corresponding arrows.

4 Incrementally building semantic request story-plans

Coherently with the main phases of a decision-making process (see Section 1), semantic request story-plans are incrementally built by DSSs in three consequent phases.

Phase1: Instantiation of the problem In the first phase, the problem part of the DSO is instantiated. This occurs when the user submits the request to the DSS via the dialogue module. That is, a module of the system processes the input selections and parameters provided by the user, and generates the instances and assertions characterizing the user decision support request. A consistency check of the input instances with the schema defined by the DSO can be performed via reasoning, to verify that the user request is compliant with the problem supported by the DSS.

Phase2: Instantiation of the data In the second phase, the actual data which will be used by the DSS to provide decision support, and generate the final conclusions, are instantiated in the A-Box, and connected to the instances describing the problem being processed. First, the DSS determines which data are relevant for the user decision

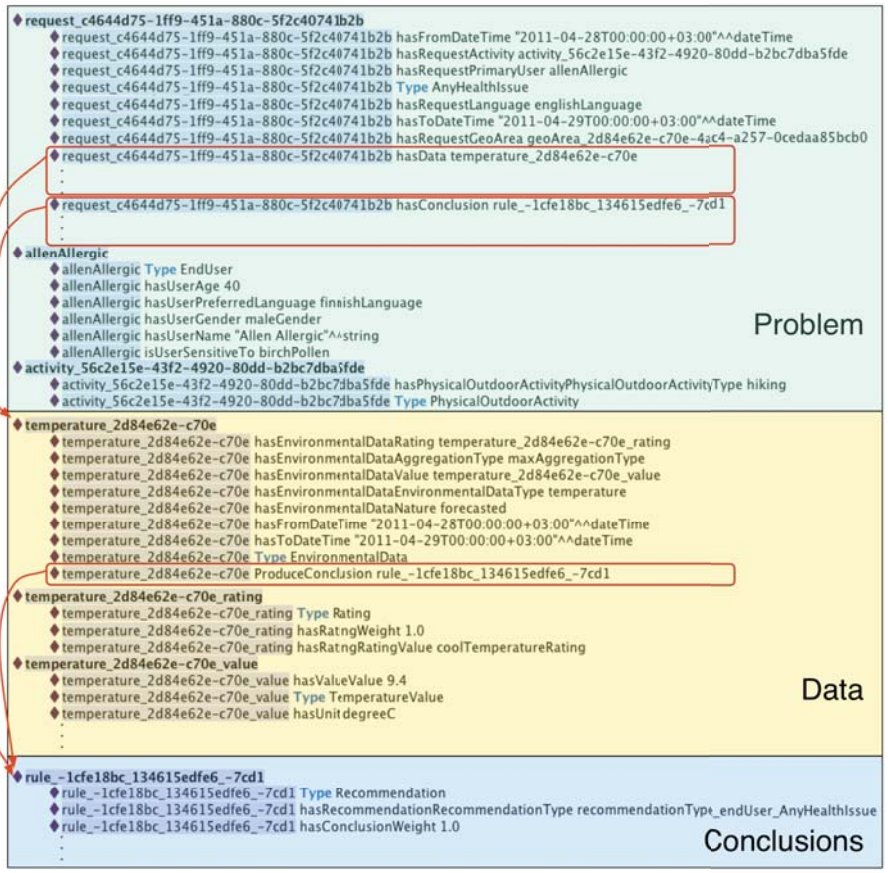


Fig. 2: An A-Box of the PESCaDO DSKB

support request submitted. For this purpose, different strategies and techniques can be exploited. In PESCaDO, we encoded in the DSO some mappings between the three parts (request, user, activity) of the Problem component, and the types of environmental data formalized in the Data component, to represent that certain environmental data are relevant for some problem aspects. These mappings, defined together with the domain experts involved in the project, are formalized as OWL hasValue restrictions on the classes of the Problem component. For instance, a restriction of the form "hasRelevantAspect **value** Rain" on the class characterizing the users sensitive to some pollen, states that data about precipitation should be retrieved and taken into consideration when providing decision support for this typology of user. By modelling the mapping this way, the environmental data types for which to retrieve data about, can be automatically determined via DL-reasoning, simply checking the new assertions inferred by the reasoner to the request, user, and activity individuals forming the current user decision request.

Once the data to be used are determined, the module of the system in charge of retrieving these data from the data sources can instantiate them in the DSKB according to the schema defined in the Data component. The connection of the ontology individuals formalizing the data with the individuals formalizing the request under processing is also instantiated (see the "hasData" assertions - first red box - Figure 2).

Phase3: Instantiation of the conclusions In the third phase, the conclusions triggered by the data according to the user decision support request are instantiated in the A-Box. The way conclusions are computed depends on the techniques for decision support implemented in the DSS. For instance, in PESCaDO we implemented a module responsible for computing conclusions which combines some complementary techniques, based on DL-reasoning and rule-based reasoning. More in details, a two-layers reasoning infrastructure is currently in place: the first layer exploits the HermiT reasoner [16] for the OWL DL reasoning services. The second layer is stacked on top of the previous layer, and implements the Jena [17] RETE rule engine, which performs the rule-based reasoning computation.

Once the conclusions are instantiated in the DSKB according to the schema defined in the Conclusions component, they are also connected to the individuals formalizing the request under processing (see the "hasConclusion" assertions - second red box - Figure 2), and to the data triggering them (see the "ProduceConclusion" assertions - third red box - Figure 2).

Each semantic request story-plan is maintained by a DSS at least for the lifetime of the processing of the request by the system. Then, the DSS can dispose of the story-plan, or it can archive it in a dedicated cases repository for other purposes (see Section 5). Note that, especially for web-based DSSs like PESCaDO, where simultaneous requests may be submitted by users to the system, several semantic request story-plans may be there at the same time in the DSKB. To manage them in parallel, the DSS can adopt an *ontology pooling* mechanism [18]: multiple in-memory ontologies (aka *pools*) are available in the system, and each decision support request submitted to the DSS is assigned exclusively to one of these pools. This solution, adopted in the PESCaDO DSS, allows to keep the size of the ontology used in each pool relatively small (on average the PESCaDO system is working with A-Boxes containing approximatively 20,000 triples), allowing to efficiently exploit the ontology also for some DL-reasoning tasks, like the ones we previously described in this section.

5 Exploitation of semantic request story-plans

In this section, we present a couple of examples where the semantic representation of a request story-plan can be further exploited to offer additional enhanced services.

5.1 Natural language generation of DSS reports

At the end of a DSS computation, the A-box associated with a decision support request contains a complete "semantic" snapshot of all the information processed and produced

by the DSS for the given request: it contains a complete description of the user request, the data relevant for the request and that were used for the decision support computation, and the conclusions and inferred content produced by the DSS together with the information on what triggered those conclusion. All this information can be used to automatically generate a text, summarizing and explaining in natural language the most relevant information to be reported to the user. In the context of an environmental DSS, like in PESCaDO, this automatically generated text, which may complement information provided by the system in graphical or tabular form, is especially appreciated by laymen, or even media corporations which may directly spread it through their communication channels.

In PESCaDO, an approach for multilingual personalized information generation from dynamically instantiated ontologies is adopted [19]. Two modules are involved in the information generation: the text planning module and the linguistic generation module. In particular, the text planning module consists of a content selection and a discourse structuring phase, both performed on a dynamically instantiated ontology (e.g., the T-Box + an A-Box of our DSKB) extended by an additional ontology module capable of representing content selection schemas and elementary discourse units. The output of the text planning module is thus an instantiated ontology enriched with information on the content selected, and the way the text should be organized. This constitutes the input of the linguistic generation module, which produces the text in the three languages supported by the system. Next we report an excerpt of the kind of text produced by the PESCaDO system by exploiting the semantic request story-plan representation

> Situation in the selected area between 08h00 and 20h00 of 07/05/2012. The ozone warning threshold value ($240g/m^3$) was exceeded between 13h00 and 14h00 ($247g/m^3$), the ozone information threshold value ($180g/m^3$) between 12h00 and 13h00 ($208g/m^3$) and between 14h00 and 15h00 ($202g/m^3$). The minimum temperature was 2°C and the maximum temperature 17°C. The wind was weak (S). There is no data available for carbon monoxide, rain and humidity.
> **Ozone warning**: ozone irritates eyes and the mucous membranes of nose and throat. It may also exacerbate allergy symptoms caused by pollen. Persons with respiratory diseases may experience increased coughing and shortness of breath and their functional capacity may weaken. Sensitive groups, like children, asthmatics of all ages and elderly persons suffering from coronary heart disease or chronic obstructive pulmonary disease, may experience symptoms. [...]

5.2 Semantic archive of request story-plans

The semantic request story-plans produced by the DSS could be archived in a semantic repository (e.g., a triple store like Virtuoso [20]), whose schema is defined by the DSO of the DSKB. This allows to build an incrementally growing archive of all the decision support requests handled by the DSS, together with the data used to process each request and the conclusions generated, as well as some feedback of the user on the decision support provided by the system.

This semantic archive of request story-plans can be exploited for several purposes, enabled by the possibility to semantically access/query its content. For instance, the

archive could be used to fine-tune the decision support strategies implemented in the DSS by querying and inspecting the requests not positively rated by the users. Similarly, in the case of DSSs implementing case-based reasoning strategies, the positively rated requests could be used to strengthen the selection of cases used for taking decisions. Furthermore, thanks to the precise semantic provided by the DSO, this archive could be exposed to the world, adopting the Linking Open Data philosophy. Therefore, the content of the archive could be further exploited by other applications or web-services, like for instance a case reuse/adaptation service which can adapt to slightly different cases the decision-making process done by the DSS, the main phases of which are tracked in each semantic request story-plan. Relevant statistics could also be produced by querying this semantic archive: e.g., in the context of PESCaDO, one may be interested in checking how frequent is the occurrence of warnings, triggered by environmental conditions, reported to sensitive users, or which type of requests are more frequently submitted by each of the various typologies of users.

6 Conclusions

In this paper we proposed to employ an ontology-based knowledge base as the main data structure in DSSs. In our approach, to each decision support request submitted to the DSS corresponds a semantic request story-plan in the knowledge base, which describes in a structured way (i) the request itself, (ii) the data relevant for the request, and (iii) the conclusions/suggestions/decisions generated by the system by processing the data. We described the possible usages and advantages offered by the proposed approach, demonstrating them in a concrete implementation for an environmental DSS, developed in the context of the PESCaDO EU project. In details, the PESCaDO DSS shows that a semantic representation of the content processed and produced by a DSS enables (i) to integrate heterogeneous sources of data available in the web (e.g., web sites, web services), (ii) to track, and to expose in a structured form to additional services (e.g., a natural language report generation service), all the content processed and produced by the DSS for each request, and (iii) to exploit logical reasoning for several of the inference steps of the DSS decision-making process. The PESCaDO DSS was positively evaluated by a team of environmental experts, who judged an appropriateness of 90% and a completeness of 87% of the content produced by the system on some exemplar scenarios considered[12].

Acknowledgements The work described in this paper has been partially funded by the European Commission under the contract number FP7-248594 (PESCaDO).

References

1. Laskey, K.B.: Decision Making and Decision Support (2006)
2. Marakas, G.M.: Decision support systems in the twenty-first century. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1999)
3. Esposito, M., De Pietro, G.: An ontology-based fuzzy decision support system for multiple sclerosis. Eng. Appl. Artif. Intell. **24**(8) (December 2011) 1340–1354

[12] A detailed description of the evaluation performed has been omitted due to the lack of space.

4. Ishizu, S., Gehrmann, A., Minegishi, J., Nagai, Y.: Ontology-driven decision support systems for management system audit. In: Proc. of the 52nd Annual Meeting of the ISSS. (2008)
5. Choraś, M., Kozik, R., Flizikowski, A., Renk, R., Holubowicz, W.: Ontology-based decision support for security management in heterogeneous networks. In: Proc. of the 5th Int. Conf. on Emerging intelligent computing technology and applications. (2009) 920–927
6. Wyner, A.: An ontology in owl for legal case-based reasoning. Artif. Intell. Law **16**(4) (December 2008) 361–387
7. Casanovas, P., Casellas, N., Vallbé, J.J.: An ontology-based decision support system for judges. In: Proceedings of the 2009 conference on Law, Ontologies and the Semantic Web: Channelling the Legal Information Flood, IOS Press (2009) 165–175
8. Ceccaroni, L., Cortés, U., Sànchez-Marrè, M.: OntoWEDSS: augmenting environmental decision-support systems with ontologies. Environmental Modelling & Software **19**(9) (2004) 785–797
9. Zagorulko, Y., Zagorulko, G.: Ontology-based approach to development of the decision support system for oil-and-gas production enterprise. In: Proceedings of the 2010 conference on New Trends in Software Methodologies, Tools and Techniques: Proceedings of the 9th SoMeT_10, Amsterdam, The Netherlands, The Netherlands, IOS Press (2010) 457–466
10. Pangjitt, T., Sunetnanta, T.: A model of ontology driven case-based reasoning for electronic issue management systems. In: Int. Joint Conf. on Computer Science and Software Engineering. (2011)
11. Saremi, A., Esmaeili, M., Habibi, J., Ghaffari, A.: O2dss: A framework for ontology-based decision support systems in pervasive computing environment. In: The 2008 Second Asia Int. Conf. on Modelling & Simulation (AMS), IEEE Computer Society (2008) 41–45
12. Wanner, L., Vrochidis, S., Tonelli, S., Moßgraber, J., Bosch, H., Karppinen, A., Myllynen, M., Rospocher, M., Bouayad-Agha, N., Bügel, U., Casamayor, G., Ertl, T., Kompatsiaris, I., Koskentalo, T., Mille, S., Moumtzidou, A., Pianta, E., Saggion, H., Serafini, L., Tarvainen, V.: Building an environmental information system for personalized content delivery. In: Proceedings of the ISESS 2011, Brno, Czech Republic, Springer (2011) 169–176
13. Smith, M.K., Welty, C., McGuinness, D.L.: Owl web ontology language guide. W3C Recommendation (2004)
14. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P.T., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E.G., den Bussche, J.V.: The open provenance model core specification (v1.1). Future Generation Comp. Syst. **27**(6) (2011) 743–756
15. Tonelli, S., Rospocher, M., Pianta, E., Serafini, L.: Boosting collaborative ontology building with key-concept extraction. In: Proceedings of the Fifth IEEE International Conference on Semantic Computing, Stanford, CA, USA (2011)
16. Shearer, R., Motik, B., Horrocks, I.: HermiT: A Highly-Efficient OWL Reasoner. In Ruttenberg, A., Sattler, U., Dolbear, C., eds.: Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 EU), Karlsruhe (October 26–27 2008)
17. HP: Jena - A Semantic Web Framework for Java. `http://jena.sourceforge.net`
18. Moßgraber, J., Rospocher, M.: Ontology management in a service-oriented architecture. In: 11th International Workshop on Web Semantics and Information Processing (WebS2012). (2012)
19. Bouayad-Agha, N., Casamayor, G., Mille, S., Rospocher, M., Saggion, H., Serafini, L., Wanner, L.: From ontology to nl: Generation of multilingual user-oriented environmental reports. In: The 17th International conference on Applications of Natural Language Processing to Information Systems (NLDB 2012), 26-28 June 2012, Groningen, The Netherlands. (2012)
20. Erling, O., Mikhailov, I.: Virtuoso: Rdf support in a native rdbms. In: Semantic Web Information Management. Springer (2009) 501–519

RING: A Context Ontology for Communication Channel Rule-based Recommender System

Miguel Lagares Lemos, Daniel Villanueva Vasquez, Mateusz Radzimski, Angel Lagares Lemos and Juan Miguel Gómez-Berbís

Departamento de Informática
Universidad Carlos III de Madrid, Spain
{miguel.lagares, daniel.villanueva, mateusz.radzimski, angel.lagares, juanmiguel.gomez}@uc3m.es

Abstract. Since a few years there has been an explosion of communication possibilities. The wide number of communication channels available nowadays brings, with no question, enormous benefits for the users, however it brings new challenges as well. The choice of the best communication channel not only depends on the characteristics related to the channel itself (QoS, rates, etc.); the user context (location, personal agenda, etc.) affects the decision as well. The combination of all the factors generates a list of variables impossible to handle by the human being. The information available about channels, together with the information that smart devices are able to extract from the user context, make possible to introduce a computing system to help the user in selecting the most appropriate communication channel.

In this paper we present RING, a context-aware ruled-based recommender system for the automatic Redirection of incomING communications based on Semantic Web, that allows users to receive any kind of communication through the best channel available depending on his context and personal preferences.

Keywords: Communication Channels, Semantic Web, Context Ontology, Rule-based Recommender.

1 Introduction

Within the telecommunications, telephony is one of the technological development fields with an ever-growing demand for services in recent years. Today, the fact of being able to establish different communications anywhere at any time is a request from users. This is the reason why, to ensure they have access to the communications channels they require, they used to be subscribed to several communication services.

Voice services currently offered by the telecommunications operators consist on fixed telephony, mobile telephony and voice over IP (VoIP). The variety of voice services, along with the infinite possibilities regarding companies, devices, etc., promotes the increment of multi-connected users; users that have multiple active connections through single or multiple voice services. As might be the case of a user with more than one SIM card, multiple telephone lines (in the same or different location) and multiple registrations to operate with VoIP. This situation makes the

user to become a “communication center” and the need for new technologies to handle the single user - multi-channel communication paradigm arises.

On the other hand, the consumer behavior has dramatically changed during last years on the way they are carrying out the communication. Twitter, Facebook, SMS, WhatsApp, BBM (BlackBerry Messenger) and many others have pushed the typical synchronous voice communication into the background. These new tools or technologies offer the possibility of an asynchronous multi-channel communication. The user will decide to use or to be addressed at one of these services depending on his/her current location, as on economic and social factors, this is the user’s information context.

The goal of the recommendation system proposed in this paper is to provide the best outbound communication channel, in order to reach the end-user based on the current user’s context. Therefore, it could make possible the integration of all the services of a given user in a unique endpoint. This can be used, for instance, for redirecting incoming calls to the service which better fulfills the requirements of the destination user at a given time. The system would offer a "virtual integration" of voice services, providing flexibility and adaptability when routing voice traffic, deciding the service which will carry out the communication in order to fit the user necessities.

Ontologies describing the user’s context can be employed to support processes focused on user features, situation and preferences, as for instance rerouting calls systems. User profiling is commonly used to support customization and flexibility [1]. RING ontology aims at creating a user’s context model for describing the current situation of a user with reliable information to be used for outbound communication channel recommendation.

The smart phones, social networks and, in general, the Information Technologies allow extracting reliable information of the user’s context, so-called Context Awareness. It deals with the fact that smart phones typically include apps (i.e. contacts or agenda related apps), multiple sensors and features that can be used to define the user’s current situation in a very accurate manner at any time. Also the social networking services make possible to improve and verify this information. In consequence, information as location, current activity, relationships, social status, camera input, etc., can be used to establish the user’s context [2].

As the user’s context information is extracted from independent and heterogeneous systems, like smart phone GPS, Outlook agenda, social networks, etc. Several agents are dealing with the context information recording and semantic annotation.

Ontologies have been proved to be an effective mean for representing knowledge related to user context. Ontologies depict concepts and relationships at a high level of abstraction, which allows human beings to understand the model representation. Well designed ontologies enable machines to use the knowledge-base and apply reasoning techniques to obtain results [1].

The remainder of this paper is organized as follows. Section 2 provides a thorough related work as a starting point. In Section 3 a motivation example is presented to introduce the problem faced in multi-channel communication. Section 4 explains and defines the RING user’s context ontology. The application of RING ontology to a communication channel rule-based recommendation system is illustrated in section 5. Finally, section 6 shows the conclusions and future work.

2 Related Work

The term Semantic Web has recently been coined to designate the next-generation Web, where content, which has semantic information, are independent of the presentation to the user, and can be processed automatically without human intervention. This creates an environment where software agents can perform tasks efficiently [3].

The Semantic Web requires a formal structure to represent the knowledge associated with the data, which is the role that ontologies are playing [4]. The two-pronged use of ontologies has the dual functions of allowing humans to grasp the meaning of any element having a well-defined vocabulary and, secondly, having formal semantics to support reasoning. In our approach, using semantic technologies as the key technology enables the data management of data generated due to the use of smart mobile phone devices (aka. Smartphones). This technology has already been used to represent knowledge in a variety of domains, such as clinical [5,6], organizational memory [7,8], knowledge management [9], bioinformatics [10] and even e-learning [11]. Ontologies allow the knowledge they contain can be reused and shared, so that their use greatly reduces the effort needed to implement expert systems. Ontologies are a key technology for the Semantic Web [12].

In this work we focus on context ontologies, they are used in context-aware systems, such as the one we propose, which exploit contextual information in order to generate recommendations or provide relevant information to the users [13]. A number of context ontologies have been developed for a variety of fields [14,15,16], there have been some efforts, as well, in the development context ontologies for mobile environments [17,18], it is worthwhile to mention the ontology in [19], where the authors present a context ontology in the mobile environment with the objective of representing the knowledge about the user regarding his interaction with mobile devices. This approach is similar to the one we present whilst this manuscript, however, our focus is not just on mobile devices, we consider all the possible communication channels that a person can use, and the contextual information that can be extracted from them.

By leveraging the Semantic Web technologies, recommendation agents enhance their understanding of users and their needs. The recommender systems can offer specific recommendations for a given user by means of personalization techniques [20]. Among them, rule-based techniques are well established and broadly used [21]. Regarding rule-based systems, a number of efforts have discussed automated extraction of rules as input to an expert system [22]. A large number of rule-based systems are also applied to very different fields such as process controlling [23], different process types optimizations [24] and treatment recommendation [25] among others. Further efforts explain the benefits of the application of such systems ([26] and [27]), from a theoretical perspective.

Finally, there are context-aware applications in the same field of this paper, such as [28] for call forwarding, [29] for reminders, [30] for social events, etc. Nevertheless, to the best of our knowledge, there is no previous work that has developed a complete ontology for all the possible communication channels to use it in a context-aware rule-based recommender system.

3 Motivating Example: Where the hell is Matt?

"Where the hell is Matt?" refers to a viral video series, where the protagonist, Matt, a young American traveler, from Westport, Connecticut, was dancing around the world with local people in front of famous landmarks and sightseeing spots.

We live in a dynamic world where people, like Matt, change their location and context in a short interval of time. Thus, the telecommunication operators have been driven to increase the amount of communication services in order to fulfill the ever-growing user's requirements. Due to the effort carried out by the telecommunication sector, very likely, Matt was able to communicate with his relatives from almost all the places where he was dancing in a multi-channel communication approach.

These services or communication channels differ from each other in their quality of the service (QoS), rates, security, networks and protocols, type of communication (synchronous or asynchronous), etc. In Matt's case, we can consider that during his trips he had to deal with communication problems, such as different time zones, high voice call rates, lack of 3G connection, etc. Therefore, probably Matt would have loved to specify the best communication channel for every situation, depending on the features offered by the different services and based on his context.

Although his friends and relatives were aware of his location due to the pictures and videos that Matt was uploading to his personal blog, it is quite sure that Matt, as any other young people, was making use of multi-channel communication, by means of different technologies and tools besides his blog, as social networks (i.e. Twitter or Facebook), instant messages programs (i.e. gChat, Microsoft Messenger or Whatsapp) VoIP services (i.e. Skype or VoipBuster), landlines in hotels and hostels, mobile phone, etc. Nevertheless, Matt went to remote places, where he was not reachable by any communication mean, due to lack of network coverage, unavailable services due to location, impossibility of accessing any endpoint, etc. In consequence, the communication was, sometimes, hard or even unfeasible. That situation caused family and relatives trying to connect through all the possible communication channels, as they did not know where, when, and how he was going to be available. In this situation, the recommender ensures that if there is at least one communication channel available, it will choose it automatically. It means that a single communication attempt, which does not succeed, is enough to be sure that Matt is not available at all.

For instance, when the network coverage was too low to maintain a regular conference over mobile phones due to several interruptions in the communication channel, Matt would have rather use SMS communication or chatting application on his smart phone or laptop. Or, when the availability of high speed internet access allowed him to use VoIP services, he would rather choose that than a costly international voice call over the landline. We can even imagine that he was in extreme situations where the only communication channel available was the phone of the chief of an African tribe; a rule in the recommender to forward all the communication to that phone would have been a great help for Matt.

In a nutshell, we face two different problems in the scenario presented here. The first one refers to the fact of managing different telephone numbers and usernames for a single user in order to be able to set up a communication, this is, know all the communication channels endpoints. And the second problem is about choosing the

service that better covers the communication requirements for the user based on destination user's context.

The overall situation could have caused communication problems among his friends and relatives to reach Matt over a communication channel. The aforementioned problem can be solved by means of a recommendation system which tells the user who establishes the communication, the best communication channel to reach the destination user at a given moment.

This recommendation system should be able to know Matt's context (location, preferences, communication channels and their characteristics, etc.). To define the context, a semantic approach, by means of an ontology, as RING ontology presented in this paper, can be used for semantic reasoning, inferring logical consequences from a set of rules to be able to make recommendations of the best outbound channel to establish the communication.

4 RING Ontology

The RING ontology captures the model of the callers' information along with necessary context, such as location, preferences, communication channels, planned events or trips. The data stored within the system (modeled after the RING ontology) is used as an input to the recommendation system that is driving the process of communication channel selection. Its main task is to ensure that the communication is successfully established with the other party, which may be a phone call, a message or a VoIP videoconference.

Figure 1 shows the overview of the process of establishing voice connection supported by the RING system. The knowledge for making necessary decision consist of 3 main parts: (i) the data model represented as lightweight ontology (providing definition of concepts, relationships and taxonomies of classes), (ii) rulebase capturing the behavior of recommendation system and (iii) profiles data set.

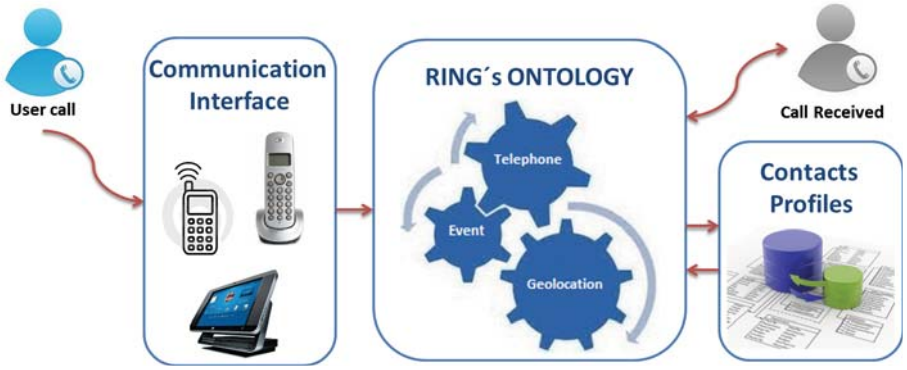


Figure 1: Functionality of RING´s Ontology.

The high-level overview of the data model is presented on Figure 2. It consist of 5 main pillars:

- Geolocation data – a history of user locations (including the most recent one) for building the location profile. It is later related with communicators used in certain places (such as using fix line at home, skype at work, etc.).
- Event schedule – user-planned events (e.g. 2-week travel to Egypt in first half of June). The schedule has priority when directing the message to the recipient, as only a subset of available message channels can be available abroad.
- Contact – list of user's contacts with their numbers (or account identifiers, depending on the communicator).
- Communicators – account list of communicators used by the user.
- Preferred communication channels – list of preferred communication channels (e.g. voice, video, message).

While description of the whole ontology is beyond the scope of the paper, we only explain part of it, to give the reader a gist of the motivation that leaded us to use a context ontology in the RING system.

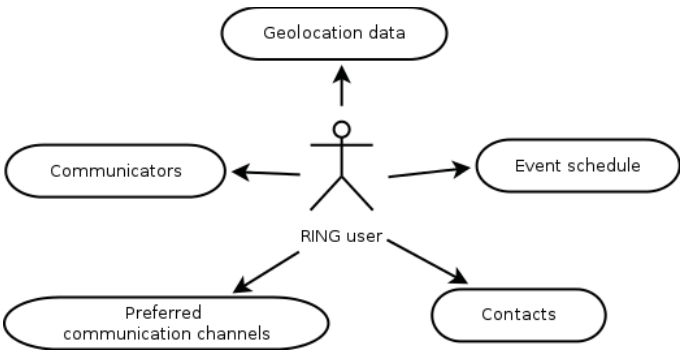


Figure 2: High-level ontology model.

Figure 3 shows a fragment of the RING ontology, which contains the event schedule part. The specific event is an instance of class event, i.e. travelWilliam that contains details of the travel (such as date, destination, etc). It is connected to the concrete user through the relation `participatesInEvent(User, Event)`. Each Event class provides additional characteristics that improve the decision process, such as additional communication means. This can be, for instance, availability of work phone during the working hours (for `Work` event type).

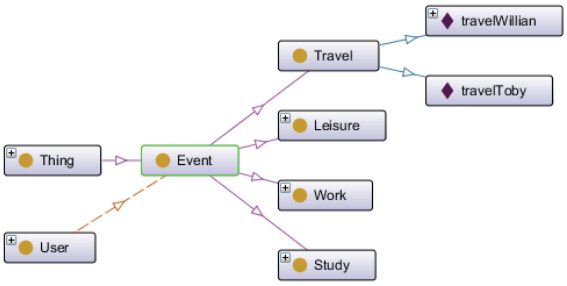


Figure 3: Example of the Event Schedule ontology fragment.

While the five RING pillars describe several aspects of the user's profile, actually they are connected through a much higher number of relationships. Those relationships represent not only facts gathered by the system (ubiquitous geolocation snapshot of user activity, call history, custom habits) or future plans (planned events), but most of the additional knowledge inferred from all the previous facts by applying the forward chaining rulebase system (based on Drools[1] rules).

5 Rule-based Recommendation System: reasoning over the RING knowledgebase

The rule-based recommendation system is being driven by general rules and user preferences. In this way, the recommendation system is using a semantic reasoning engine to be able to infer logical consequences from RING context-aware ontology. The materialized factbase is later augmented with custom, domain-specific knowledgebase in a form of if-then rules (production rule system). Inferred facts (in a form of new data relations) are crucial for communication channel recommendation. The knowledge-based systems with large structures of concepts and rules are being used extensively in a wide variety of applications.

The goal of the recommendation system is to integrate all the communication services of a given user in a unique endpoint and to provide the channel which better fulfills the requirements of the user, based on user's context, at a given time to carry out the communication. The purpose of the integration of the services is to provide to the users with a unique access point to establish communication. Thus it allows other users to contact each other in a straightforward manner and in consequence simplifying the system and increasing the probability to communicate with the end-user. In other words, tie the communication to a user instead to a device, by generating precise recommendations. Therefore the system offers a "virtual integration" of a set of communication channels, providing the service, which will carry out the communication in order to fit the user necessities.

[1] See: http://www.jboss.org/drools/

As explained before, the call is redirected depending on the user's context (location, agenda, etc.) modeled in RING ontology. With this aim, a set of agents deals with external information systems in order to extract all the relevant information to populate the RING ontology. Agents are heterogeneous and independent from each other; they obtain information from different sources, such as smartphones, social networks, and the like. The information collected by the agents is used in the annotation tool to instantiate the knowledge base, which contains the instances of the reference ontology.

The user preferences are a set of rules determined by the user for driving the rule-based recommendation system. These preferences establish a policy in the system. These preferences, used in a later stage to generate the rules, refer to schedules, priorities, costs, location, etc.

6 Conclusions and Future Work

In this paper, we have presented RING, a context ontology for communication channel rule-based recommender system. The representation of user's context data in a semantic modeling in conjunction with the rule-based reasoning are the core parts of the system proposed, which aims to provide recommendation on the best channel to carry out the desired communication.

We truly believe that RING can be integrated in any platforms for rerouting communication requests over several outbound communication channels, to provide recommendations depending on the current situation of the recipient user. Generating an added value to the platform by offering flexibility and adaptability for establishing communications.

Future research lines in the context of RING will analyze qualitative and quantitative aspects for a complete evaluation of the system, a real use case from the project GECALLIA (A Geolocation System for Call Routing based on Artificial Intelligence and Semantic Web) will be used for that purpose.

We will consider as well using a Private Branch Exchange (PBX) or a similar system to carry out call rerouting based on user's context.

Some changes in the proposed system to study the performance of the recommender will be investigated, the use of neural networks instead of rule-based recommender is an appealing option. Another option would be to introduce parameters related to the behavior of the user in different situations, in this way the system would be able to learn with the actions of the user in order to provide more adaptability and flexibility for each user of the system

Acknowledgments

This work was supported by the Spanish Ministry of Industry, Tourism, and Commerce under the projects GECALLIA (TSI-020100-2011-244), ENERFICIENCY (TSI-020400-2011-56) and (SEMOSA TSI-020400-2011-51). It

was also supported by the Spanish Ministry of Science and Innovation under the projects TRAZAMED (IPT-090000-2010-007) and FLORA (TIN2011-27405).

References

1. M. Golemati, A. Katifori, C. Vassilakis, G. Lepouras, and C. Halatsis, "Creating an ontology for the user profile: Method and applications," in Proceedings of the First RCIS Conference, 2007, pp. 407–412.
2. M. Bloice, M. Kreuzthaler, K. Simonic, and A. Holzinger, "On the paradigm shift of search on mobile devices: some remarks on user habits," HCI in Work and Learning, Life and Leisure, pp. 493–496, 2010.
3. T. Lee, J. Hendler, O. Lassila et al., "The semantic web," Scientific American, vol. 284, no. 5, pp. 34–43, 2001.
4. B. Xiao and I. Benbasat, "E-commerce product recommenda- tion agents: Use, characteristics, and impact," Mis Quarterly, vol. 31, no. 1, pp. 137–209, 2007.
5. Y. Shahar and M. Musen, "Knowledge-based temporal abstrac- tion in clinical domains," Artificial intelligence in medicine, vol. 8, no. 3, pp. 267–298, 1996.
6. S. Schulz, M. Romacker, G. Faggioli, and U. Hahn, "From knowledge import to knowledge finishing: automatic ac- quisition and semi-automatic refinement of medical knowl- edge," in Proceedings of the Banff Knowledge Acquisition for Knowledge-Based Systems Workshop. Citeseer, 1999.
7. R. Dieng, O. Corby, A. Giboin, and M. Ribiere, "Methods and tools for corporate knowledge management," International journal of human-computer studies, vol. 51, no. 3, pp. 567– 598, 1999.
8. D. Schwartz, "When email meets organizational memories: ad- dressing threats to communication in a learning organization," International journal of human-computer studies, vol. 51, no. 3, pp. 599–614, 1999.
9. Y. Sure and R. Studer, "A methodology for ontology-based knowledge management," Towards the Semantic Web, pp. 33– 46, 2003.
10. Stevens, R., Wroe, C., Lord, P., and Goble, C., "Ontologies in bioinformatics". In Stefan Staab and Rudi Studer, editors, Handbook on Ontologies, pp. 635-657. Springer, 2003.
11. J. Brase and W. Nejdl, "Ontologies and metadata for elearning," Springer-Verlag, 2003.
12. J. Davies, D. Fensel, and F. Van Harmelen, Towards the semantic web. Wiley Online Library, 2003.
13. G. Chen and D. Kotz, "A survey of context-aware mobile computing research," Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover, NH, Nov 2000.
14. T. Gu, X. Wang, H. Pung, and D. Zhang, "An ontology-based context model in intelligent environments," in Proceedings of communication networks and distributed systems modeling and simulation conference. Citeseer, 2004, pp.270–275.
15. T. Strang, C. Linnhoff-Popien, and K. Frank, "Cool: A context ontology language to enable contextual interoperability," in Distributed applications and interoperable systems. Springer, 2003, pp. 236–247.
16. D. Preuveneers, J. Van den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, and K. De Bosschere, "Towards an extensible context ontology for ambient intelligence," Ambient intelligence, pp. 148–159, 2004.

17. P. Gutheim, “An ontology-based context inference service for mobile applications in next-generation networks,” IEEE Communications Magazine, vol. 49, no. 1, pp. 60–66, 2011.
18. P. Korpipaa, J. Hakkila, J. Kela, S. Ronakainen, I. Kansala, “Utilising context ontology in mobile device application personalisation,” in Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia. ACM, 2004, pp. 133–140.
19. M. Poveda Villalon, M. Suarez-Figueroa, R. Garcia-Castro, and A. Gomez-Perez, “A context ontology for mobile environments,” 2010.
20. G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” Knowledge and Data Engineering, IEEE Transactions on, vol. 17, no. 6, pp. 734–749, 2005.
21. M. Pazzani and D. Billsus, “Content-based recommendation systems,” The adaptive web, pp. 325–341, 2007.
22. S. Tsumoto, “Automated extraction of medical expert system rules from clinical databases based on rough set theory,” Information Sciences, vol. 112, no. 1, pp. 67–84, 1998.
23. J. Bernard, “Use of a rule-based system for process control,” Control Systems Magazine, IEEE, vol. 8, no. 5, pp. 3–13, 1988.
24. A. de Geus and W. Cohen, “A rule-based system for optimizing combinational logic,” Design & Test of Computers, IEEE, vol. 2, no. 4, pp. 22–32, 1985.
25. G. Guyatt, J. Sinclair, D. Cook, and P. Glasziou, “Users’ guides to the medical literature: XVI. how to use a treatment recommendation,” Journal American Medical Association, vol. 281, pp. 1836–1843, 1999.
26. R. Hillestad, J. Bigelow, A. Bower, F. Girosi, R. Meili, R. Scoville, and R. Taylor, “Can electronic medical record systems transform health care? potential health benefits, savings, and costs,” Health Affairs, vol. 24, no. 5, pp. 1103–1117, 2005.
27. K. Kawamoto, C. Houlihan, E. Balas, and D. Lobach, “Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success,” Bmj, vol. 330, no. 7494, p. 765, 2005.
28. R. Want, A. Hopper, V. Falcao, and J. Gibbons, “The active badge location system,” ACM Transactions on Information Systems (TOIS), vol. 10, no. 1, pp. 91–102, 1992.
29. P. Ludford, D. Frankowski, K. Reily, K. Wilms, and L. Terveen, “Because I carry my cell phone anyway: functional location-based reminder applications,” in Proceedings of the SIGCHI conference on Human Factors in computing systems. ACM, 2006, pp. 889–898.
30. D. Quercia, N. Lathia, F. Calabrese, G. Di Lorenzo, and J. Crowcroft, “Recommending social events from mobile phone location data,” in Data Mining (ICDM), 2010 IEEE 10th International Conference on. IEEE, 2010, pp. 971–976.
31. V. Krotov and I. Junglas, “Mobile technology as an enabler of organizational agility,” in International Conference on Mobile Business (ICMB). IEEE, 2006, pp. 20–20.
32. G. Lorenz, T. Moore, G. Manes, J. Hale, and S. Shenoi, “Securing ss7 telecommunications networks,” in Workshop on Information Assurance and Security, vol. 2. Citeseer, 2001, p. 1115.