Mobile Movie Recommendations with Linked Data

Vito Claudio Ostuni, Giosia Gentile, Tommaso Di Noia,
Roberto Mirizzi, Davide Romito, Eugenio Di Sciascio

Polytechnic University of Bari, Italy
{ostuni,g.gentile,mirizzi,d.romito}@deemail.poliba.it,
{t.dinoia,disciascio}@poliba.it

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 graph information within DBpedia, one of the best-known datasets publicly available in the Linked Open Data (LOD) project.

Keywords: Context-aware Recommender Systems, DBpedia, Linked Data, Movie Recommendation

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 does 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 [23]. 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 graph-based 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) `Cinemappy` [19]: 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 in mobility

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 [20], 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 the user and his/her friends love romantic movies but this is not the case of his/her partner. So, it would be fine if a movie recommender engine suggested romantic movies when the user is with his/her friends and comedies when he/she is with his/her partner.

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***: usually users prefer locations where they 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 Feeding a Content-Based RS using the Web of Data

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[1] 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 content-based (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[2] 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.

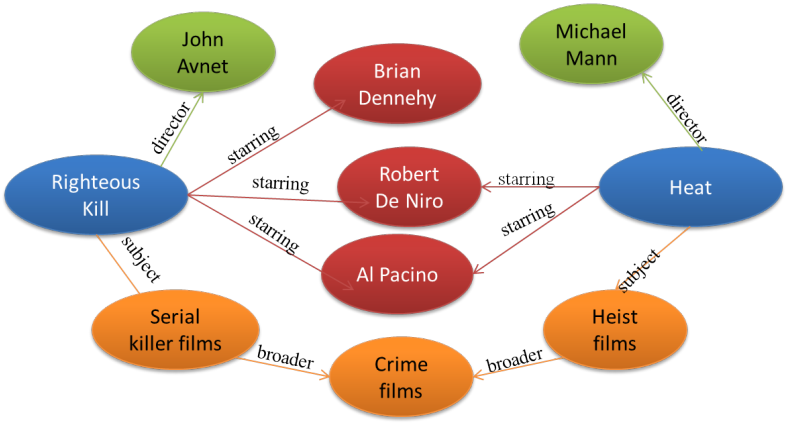


Fig. 1. A sample of an RDF graph related to the movie domain.

[1] http://www.w3.org/RDF/
[2] http://www.w3.org/TR/rdf-sparql-query/

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.

Figure 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. In Section 4.1 we will provide more details on the use of DBpedia by the recommendation engine.

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*[3] or *Trovacinema*[4]. On the other side, the CB part of the recommendation engine exploits the DBpedia graph structure for computing 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 Figure 2. All the data about the descriptions of

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

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

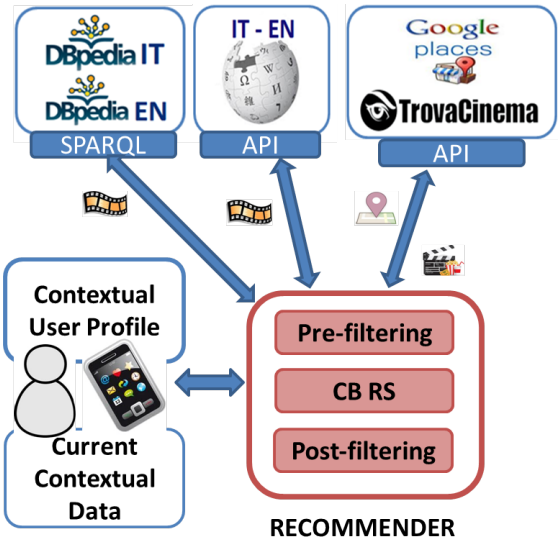


Fig. 2. System architecture.

the movies are extracted via the `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 *Google Geocoding API*[5]. 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*[6]. These lists are created considering the relative distances between theaters and POIs and they use the *Google Distance Matrix API*[7]. 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

[5] `https://developers.google.com/maps/documentation/geocoding/`
[6] `https://developers.google.com/places/documentation/`
[7] `https://developers.google.com/maps/documentation/distancematrix/`

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[8], 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. 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. For the sake of completeness we briefly report here the main elements of the approach. In order to compute the similarities between movies, the Vector Space Model (VSM) [24] is adapted to an `LOD`-based setting. In VSM non-binary weights are assigned to index terms in queries and in documents (represented as sets of terms), and are used to compute the degree of similarity between each document in a collection and the query. In [12] the VSM, usually used for text-based retrieval, is

[8] `http://youtube-global.blogspot.it/2009/09/five-stars-dominate-ratings.html`

adapted in order to to deal with RDF graphs. In a nutshell, the whole RDF graph is represented as a 3-dimensional matrix where each slice refers to an ontology property and represents its adjacency matrix. A component (i.e. a cell in the matrix) is not *null* if there is a property that relates a subject (on the rows) to an object (on the columns). Given a property, each movie is seen as a vector, whose components refer to the *term frequency-inverse document frequency* TF-IDF (or better, in this case, *resource frequency-inverse movie frequency*). For a given slice (i.e. a particular property), the similarity degree between two movies is the correlation between the two vectors, and it is quantified by the cosine of the angle between them. All the nodes of the graph are represented both on the rows and on the columns of the matrix. A few words need to be spent for the properties `dcterms:subject` and `skos:broader` which are very popular, e.g., in the `DBpedia` dataset. As also shown in Figure 1, every movie is related to a category by the property `dcterms:subject` which is in turn related to other categories via `skos:broader` organized in a hierarchical structure. To the purpose of the recommendation, `skos:broader` is considered as *one-step transitive*. We will explain this notion with the aid of a simple example. Suppose to have the following RDF statements:

```
dbpedia:Righteous_Kill
    dcterms:subject dbpedia:Category:Serial_killer_films .
dbpedia:Category:Serial_killer_films
    skos:broader dbpedia:Category:Crime_films .
```

Starting from `dbpedia:Category:Serial_killer_films` we have that `dbpedia:Category:Crime_films` is at a distance of one step. Hence, by considering a *one-step transitivity*, we have that:

```
dbpedia:Righteous_Kill
    dcterms:subject dbpedia:Category:Crime_films .
```

is inferred by the original statements.

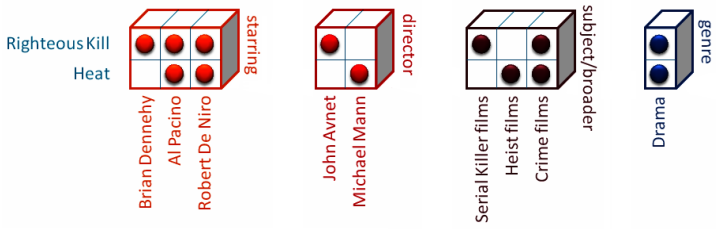


Fig. 3. Matrix representation of the RDF graph of Figure 1.

By looking at the model, we may say that: (1) the matrix is very sparse; (2) properties are considered as independent with each other (there is no `rdfs:subPropertyOf`

relation); (3) the focus is on discovering the similarities between movies (or in general between resources of the same `rdf:type` and not between each pair of resources). Based on the above observations, the matrix slices can be decomposed into smaller matrices where each matrix refers to a specific `RDF` property, as shown in Figure 3. In other words, for each matrix, the rows represent somehow the *domain* of the considered property, while the columns its *range*. For a given property, the components of each row represent the contribution of a resource (i.e. an actor, a director, etc.) to the corresponding movie. With respect to a selected property p, a movie m is then represented by a vector containing all the terms/nodes related to m via p. As for classical Information Retrieval, the index terms $k_{n,p}$, that is all the nodes n linked to a movie by a specific property p, are assumed to be all mutually independent and are represented as unit vectors of a t-dimensional space, where t is the total number of index terms. Referring to Figure 3, the index terms for the *starring* property are *Brian Dennehy*, *Al Pacino* and *Robert De Niro*, while $t = 3$ is the number of all the actors that are objects of a triple involving *starring*. The representation of a movie m_i, according to the property p, is a t-dimensional vector given by $\overrightarrow{m_{i,p}} = (w_{1,i,p}, w_{2,i,p}, ..., w_{t,i,p})$, where $w_{n,i,p}$ is a non-negative and non-binary value representing the weight associated with a term-movie pair $(k_{n,p}, \overrightarrow{m_{i,p}})$. The weights $w_{n,i,p}$ adopted in the model are TF-IDF weights. More precisely, the TF ($f_{n,i,p}$) is the frequency of the node n, as the object of an `RDF` triple having p as property and the node i as subject (the movie). Actually, this term can be either 0 (if i is not related to n via p) or 1, since two identical triples can not coexist in an `RDF` graph. As for classical information retrieval, the IDF is computed as the logarithm of the ratio between M, that is the total number of movies in the collection, and $a_{n,p}$, that is the number of movies that are linked to the resource n, by means of the predicate p. As an example, referring to Figure 3, for the *starring* property, and considering $n = AlPacino$, then $a_{AlPacino,starring}$ is equal to 2, and it represents the number of movies where *Al Pacino* acted. Relying on the model presented above, each movie can be represented as a $t \times P$ matrix, where P is the total number of selected properties. If we consider a projection on a property p, each pair of movies, m_i and m_j, are represented as t-dimensional vectors. The degree of similarity between m_i and m_j with respect to p is evaluated as the correlation between the vectors $\overrightarrow{m_{i,p}}$ and $\overrightarrow{m_j, p}$. More precisely, the correlation is computed as the cosine of the angle between the two vectors:

$$sim^p(m_i, m_j) = \frac{\sum_{n=1}^{t} w_{n,i,p} \cdot w_{n,j,p}}{\sqrt{\sum_{n=1}^{t} w_{n,i,p}^2} \cdot \sqrt{\sum_{n=1}^{t} w_{n,j,p}^2}}$$

The method described so far is general enough and it can be applied when the similarity has to be found between resources that appear as subjects of `RDF` triples. When the resources to be ranked appear as objects of `RDF` triples, it is simply a matter of swapping the rows with the columns in the matrices of Figure 3 and applying again the same algorithm. Lastly, when two resources are directly related by some specific properties (as the case of the property `dbpedia:subsequentWork`), it is sufficient to operate a matrix transformation

to handle this case in the same way as done so far. In the following we will see how to combine such similarity values with a user profile to compute a content-based recommendation. 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 $profile(u, cmp)$. A weight α_p is assigned to each property representing its worth with respect to the user profile. In order to compute these weights, supervised machine learning techniques are adopted. In particular we use a genetic algorithm to find the optimal weights. We train our model on `MovieLens`[9], a popular dataset in the movie domain.

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 with $\tilde{r}_{PreF}(u_{cmp}, m_i)$ to obtain a single score:

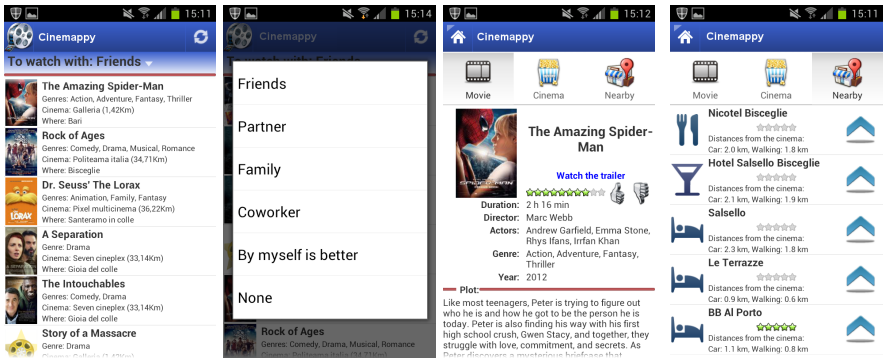
$$\tilde{r}(u_{cmp}, m_i) = \beta_1 \times \tilde{r}_{PreF}(u_{cmp}, m_i) + \beta_2 \times \frac{(\mathtt{h} + \mathtt{c} + \mathtt{cl} + \mathtt{ar} + \mathtt{ap})}{5}$$

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.

[9] `http://www.grouplens.org/node/12`

4.2 Implementation

`Cinemappy` has been implemented as a mobile application for Android smartphones[10]. When the user starts the application, `Cinemappy` displays a list of



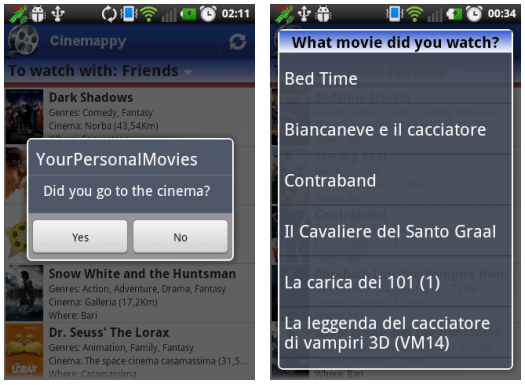
(a) (b) (c) (d)

Fig. 4. Some screenshots of `Cinemappy`.

movies according to the current contextual user profile (Figure 4(a)). The user can choose their current companion from a list of different options thus enabling their micro-profile (Figure 4(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* (Figure 4(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 (Figure 4(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

[10] `https://play.google.com/store/apps/details?id=it.sisinflab.lod.mobile.cinemappy&hl=en_GB`



(a) (b)

Fig. 5. Location-Based Rating Acquisition Service.

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 (Figure 5(a)) and and if a positive answer ensues, the user can rate one of the movies featured in that cinema (Figure 5(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.

5.1 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 [21] 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 [23], 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 [25] 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.

5.2 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* [22] 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] a model-based approach and a memory-based one to compute CB recommendations are presented leveraging `LOD` datasets. Several strategies are described and compared 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 support of PON01_00850 ASK-HEALTH project.

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. V. C. Ostuni, T. Di Noia, R. Mirizzi, D. Romito, and E. Di Sciascio. Cinemappy: a context-aware mobile app for movie recommendations boosted by dbpedia. In *1st International Workshop on Semantic Technologies meet Recommender Systems & Big Data (SeRSy 2012)*, volume 919. CEUR-WS, 2012.
20. 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.
21. 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.
22. 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.

23. F. Ricci. Mobile recommender systems. *J. of IT & Tourism*, 12(3):205–231, 2011.
24. G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, November 1975.
25. 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.