A Reusable Methodology for the Instantiation of Social Recommender Systems

Lara Quijano-Sánchez, Juan A. Recio-Garcia and Belén Díaz-Agudo  
Department of Software Engineering and Artificial Intelligence,  
Universidad Complutense de Madrid, Spain  
lara.quijano@fdi.ucm.es, jareciog@fdi.ucm.es, belend@fdi.ucm.es

Abstract—Social recommender systems exploit the social knowledge available in social networks to provide accurate recommendations. However, their instantiation is not straightforward due to its complexity. To alleviate this development complexity, we propose a methodology based on templates that conceptualize the behavior of such applications and can be reused to create several social recommender applications in social networks. This development methodology comprises not only templates but also a generic architecture named ARISE and a collection of software components that provide the required functionality. We prove that our social templates speed up and facilitate the development process, and demonstrate the viability of our generic architecture in two different case studies.

Keywords—Templates; Generic Architecture; Social recommenders

I. INTRODUCTION

Recommender systems are born from the idea of suggesting automatically items to users that they may find appealing (e.g. see [2] for an overview). When users of such systems operate not individually but in groups, recommendations to groups of users appear [11]. In the literature, (e.g. [8], [10], [14]) it was shown that using social network information in addition to feedback data (e.g. ratings) can significantly improve recommendations’ accuracy.

Social systems by their definition encourage interaction between users and both online content and other users, thus generating new sources of knowledge for recommender systems. Web 2.0 users explicitly provide personal information and implicitly express preferences through their interactions with others and the system (e.g. commenting, friending, rating, etc.). These various new sources of knowledge can be leveraged to improve recommendation techniques and develop new strategies which focus on social recommendation [16].

Our previous work [21], [18] showed an improvement in the accuracy of recommendations for groups by taking into account social information from the group, namely, the personality of the users in the group, and the strength of their connections which we refer to as their trust. These techniques and their associated algorithms have been compiled in an organized generic architecture named ARISE (Architecture for Recommendations Including Social Elements) that can be instantiated into any kind of social recommender systems that take into account the personality composition of the group and the social connections between the group members. Finally, we have applied our method of social recommendations to an instantiation of our model in a real-life scenario: HappyMovie [19], which is a particular instantiation of our generic ARISE architecture for the movie recommendation domain in the social network Facebook. HappyMovie has served us as a use case and experimental environment where we have evaluated our ARISE architecture and our social recommendation methods with real data.

In this paper we present the development methodology to reuse the ARISE architecture and the social recommendation methods. More precisely, we propose a semi-automatic way of designing social recommender systems through social templates in a CBR way. Case-based reasoning (CBR) has been used in recommender systems before (e.g. [24]) and explicit parallels between CBR and recommenders have been drawn (e.g. [17]). Templates are explicit formalizations that abstract the behaviour of a recommender system and can be reused to instantiate custom applications through, for example, the tools provided by the COLIBRI STUDIO platform [23].

The first contribution of this paper is to prove the usability and acceptance of the set of social templates that we have designed for the construction of social recommender applications. We want to prove that when developers use these templates their work is quickened and facilitated thus they prefer to use them than starting a whole project from scratch.

The second goal of this paper is to test the ARISE platform and the social templates here proposed by building a second social recommender, that belongs to a different domain from the one that we had already tested (movies): HappyShopping. HappyShopping is a social recommender system that provides clothing recommendations to people connected through social networks, i.e. clothing recommendations to Facebook users. One of the main ideas followed in our social recommendation method and adopted in the development of our applications HappyMovie and HappyShopping is that everyone is influenced by their social context. Social media highly influences our shopping, relationships, and education. Several researchers study the impact of social media in our lives [5]. The social context, refers to the immediate physical
and social setting in which people live. It includes the culture that the individual was educated or lives in, and the people and institutions with whom they interact. Circumstantial life events, influences, and surroundings can further change our behaviour [3].

The paper runs as follows. Section II presents an overview of our architecture ARISE; Section III describes our set of social templates as an intermediate step between ARISE and any social application that can be built following ARISE’s structure; Section IV describes HappyShopping as a case study of instantiation of the templates and gives a preliminary evaluation of the effort and viability of using ARISE and our templates; and Section V concludes and presents some ideas for future work.

II. GENERIC ARCHITECTURE FOR GROUP RECOMMENDERS USING SOCIAL ELEMENTS

ARISE\(^1\) is a theoretical organization of the modules required to build social recommenders [20]. This architecture allows us to simulate in a more realistic way the decision making process performed by people when choosing an item of their liking. Note that this social architecture is viable not only for group recommender systems, as it was used when building HappyMovie [19], but also for individual recommender systems, as it has been used when building HappyShopping.

The common and key factor in all the different types of recommenders that can be built in all sort of domains using this generic architecture is the inclusion of social elements. These social elements, that in our social recommendation method are the personality and trust factors, define each person (our users involved in the recommendation processes) as a potentially influenced component of a social community or group determined by the environment, in most cases social networks, s/he belongs to. In our social method, we have simulated people’s behaviour based on the idea that the relationship between individuals and their networks of people directly influence their lives [5].

The architecture ARISE is represented in Figure 1. We can see that it is divided in seven different modules: personality, trust, memory & satisfaction, individual estimation, explicit individual preferences, product data, and the ARISE module itself (which is only necessary when using the architecture for group recommendations as we will later explain). Below we summarize each of the modules (further details can be found in [20]):

1) **Personality Module**: This module fulfils the task of obtaining a value that represents the personality of each user. This personality value, \( p_u \), fits within a range of \((0,1]\), 0 being the reflection of a very cooperative person and 1 the reflection of a very selfish one.

2) **Trust Module**: This module fulfils the task of obtaining the trust values, \( t_{u,v} \), between every user \( u \) and \( v \) that belong to a common social environment or group. Note that \( t_{u,v} \in (0,1] \), 0 being the reflection of a person not to be trusted and 1 the reflection of a highly trusted one.

3) **Memory & Satisfaction Module**: This module stores all the recommendations that have been made for every user and every group [1]. This avoids repeating past recommendations and also ensures a certain degree of fairness in the long run. We believe that this is a necessary step when providing a whole set of fair recommendations. This way, if one user accepts a proposal that s/he was not interested in, next time s/he will have some kind of priority in the recommendation process.

4) **Individual Estimation Module**: This module is in charge of computing individual predictions, \( \text{pred}(u,i) \), for each user \( u \) and each item \( i \) in the catalogue. The individual predictions, or recommendations, consist on a basic building block of the architecture as our recommendation approach predicts the rating that each user would assign to every item in the catalogue and later, if used for group recommender applications, these estimated ratings are aggregated to obtain a global prediction for the group.

5) **Explicit Individual Preferences Module**: This module obtains information about the user, which is required to predict the rating for a new item. Commonly, it just consists of ratings given to some products in the catalogue.

6) **Product Data Module**: This module obtains the catalogue of products to be recommended.

7) **ARISE Module**: This module is only needed when using the architecture for group recommender systems. It combines all the information provided by the rest of the modules using our social group recommendation methods and offers a recommendation for the group. Space limitations preclude a detailed description of the social group recommendation process but it is described in [21], [20].

III. TEMPLATES

As we have introduced, our goal with the development of our social templates is to create an intermediate step between our generic architecture ARISE and any social application that can be built following ARISE’s structure.

In order to facilitate the architecture instantiation process, we propose a case-based approach where the designer retrieves a system (i.e. our social templates) from a library

\(^1\)Architecture for Recommenders Including Social Elements
(case base) of previously designed CBR systems (i.e. social recommenders) and, if needed, adapts it by adding, removing or substituting components in the selected system. Retrieval and adaptation of systems are possible through the use of semantic templates that have been previously abstracted from available systems. Each template is a generalization of several CBR systems and also include semantic annotations from human experts. Templates store the control flow of CBR systems, conceptualizing their behavior, and including the concepts and constraints required to model a number of related systems [22].

The templates that we have created for our social recommendation method are composed by tasks that identify the steps of the recommender system and methods that solve each task with a particular implementation. In this paper we present generic social templates that provide a high-level view of a set of final social templates. These templates are composed by generic tasks and simple tasks. Generic tasks encapsulate sequences of simple tasks. Depending on the decomposition of each generic task into sequences of simple tasks, we obtain several final templates.

In order to design our templates we have used the COLIBRI STUDIO Integrated Development Environment⁡ that facilitates the creation of templates for CBR systems.

A. ARIS E’s Social Templates

COLIBRI STUDIO comprises a set of tools to instantiate CBR and recommender applications based on the jCOLIBRI framework. This framework provides the basic building blocks required to easily develop such systems. Because jCOLIBRI is aimed at developer users, COLIBRI STUDIO alleviates the programming tasks and provides several graphical tools that can be used to generate automatically CBR systems. The generation of CBR applications in COLIBRI STUDIO is guided by the COLIBRI development process that proposes the reuse of existing designs –templates– of CBR applications and its adaptation to the concrete target system. These templates must follow the conceptual organization of CBR systems stated by the jCOLIBRI framework: a precycle where cases and reasoning resources are loaded; the CBR reasoning cycle itself; and an eventual postcycle step where initial resources are released.

Templates represent, in a conceptual level, the behaviour of a family of CBR systems (such as our social recommenders) but do not provide the functionality required to build them. This functionality is provided by the components in jCOLIBRI or developed on-demand. This way, the templates for building social group recommenders require the components that provide its functionality, and these components are organized in the ARIS E’s modules. We will see that some of the different tasks of the templates correspond to some of ARIS E’s modules. Note that each task can be implemented by several different methods, we will here present some of the possible methods that can be used to perform each task. Most of them are already implemented in jCOLIBRI and therefore will facilitate the process of building a new application by using our social templates as we will see in Section IV.

Figure 2 left, shows the pre-cycle template. The pre-cycle template is formed by the following tasks:

1) ObtainGroup: Consists of obtaining the id of each user \( u \in G \), being \( G = \{ u : 1 \ldots n \} \) the active group of users and \(|G| > 1\). The active group of users \( G \) is defined for social group recommender systems as the people that intend to realize an activity together, and for social individual recommender systems as the people who belong to the circle of trusted people in the social environment of the user receiving the recommendation. For both options, the group is defined in the framework of social networks. Some of the different methods that can be used to obtain \( G \) are:

- Through the creation of an event to perform an activity together (social group recommender systems).
- Calculating the group of closest friends in the social network (social individual recommender systems). To do so, the method obtains a trust value (as we will explain bellow in the ObtainTrustFactors task) with all the user’s friends in Facebook that also use the application being implemented.

2) LoadGroupHistory: (Corresponds to the Memory & Satisfaction module in ARIS E) Assume a case base \( CB \) in

⁡http://www.colibrichrstudio.net
which each case \( c \in CB \) represents a previous recommendation event. This task consists of retrieving the case \( c \) that corresponds to the active user \( u \) or group of users \( G \). Note that this task is optional and can be skipped if developers do not want to build a system with memory of past recommendations.

3) **ObtainSocialFactors**: Consists of a generic task that encapsulates the following subtasks:

- **ObtainUsersPersonality**: (Corresponds to the Personality module in ARISE) Consists of obtaining the personality of each user \( u \), denoted \( p_u \), by making users complete a personality test on registration with the recommender. This task can be fulfilled by the following methods:
  
  - The Thomas-Killmann Conflict Mode Instrument (TKI) [27] that proposes 30 situations where the user has to think about how s/he will react. (e.g. as used in [18])
  
  - TKI’s alternative movie metaphor, that consists of displaying two well known movie characters with opposite personalities for each of five possible categories. One character represents the essential characteristics of one category, while the other one represents all the opposite ones. What the user has to do is to choose which of each pair of characters s/he feels more identified with by simple moving an arrow (e.g. as used in HappyMovie [19] and in HappyShopping).
  
  - Any other personality test from which the \( p_u \) can be defined as a single numeric value.

- **ObtainTrustFactors**: (corresponds to the Trust module in ARISE) Consists of obtaining the trust \( t_{u,v} \) between users \( u \) and \( v (u \neq v \in G) \). It can be based on distance in the social network, the number of friends in common, relationship duration, and so on.

4) **LoadCases**: Consists of obtaining all the items \( i \) in the domain catalogue \( I = \{ i : 1 \ldots m \} \).

5) **ObtainIndividualPreferences**: (Corresponds to the Explicit Individual Preferences module in ARISE) Consists of obtaining the ratings \( r_{u,i} \) that each user \( u \) in \( G \) assigns to items \( i \) in \( I \). Ratings are on a numeric scale, e.g. 1 = terrible and 5 = excellent.

Now, we continue the templates explanation with the cycle template shown in the Figure 2 right. This template is principally designed for its use in the process of building social group recommender systems, however it can also be used for social individual recommenders leaving the last 4 tasks unimplemented. The cycle template is formed by the following tasks:

6) **ObtainRecommendableCases**: (Corresponds to the Product Data module in ARISE) Consists of obtaining all the candidate target items \( i \) in the recommendation catalogue \( T = \{ i : 1 \ldots t \} \).

7) **Scoring**: (Corresponds to the Individual Estimation module in ARISE) Consists of obtaining predicted ratings \( \text{pred}(u,i) \) for each active user \( u \in G \) and target item \( i \in T \). Some of the different methods that can be used to implement this task are:

- **Collaborative recommenders** [7], [12], [9], use the ratings already assigned by the users to several products. Users are selected according to their similarity with the individual receiving the recommendation (by comparing the ratings given to the products). Most similar users are used as predictors and their ratings are combined to estimate the rating that the target user would assign to a new product.

- **Content-based recommenders** [13], compare each item in the catalogue with the items already rated by the target user. Then the ratings of the most similar rated items are combined to provide a recommendation.

- **Hybrid recommenders** [4], that are a combination of the two previous ones.

- **Asking other \( G \) users to give an estimated rating for the product \( i \) [6], this method relies heavily on explicit feature-level feedback from users.

- **Influence based recommenders** [21], [18], modify the non-social predictions \( \text{pred}(u,i) \) obtained with one of the above methods with the personality and trust factors. We will detail this method as it is the one that is used for HappyShopping’s recommendations as we will see in Section IV. It supposes that the user may modify her/his preference for an item depending on the preferences given by her/his friends to the same item. For example, if our rating for an item is 3 and our friend has a 5 rating for the same item, we could think on modifying our rating to 4. Depending on the trust in this friend, we decide the level of variation for our rating (i.e. 3.5 if the trust is low, and 4.5 if trust is high). Furthermore, the variation of our rating also depends on our personality. If we have a strong personality (high personality value) we will not be willing to change our rating, but if we have a weak personality (low value) we could be easily influenced by other users.

The method combines the personality and trust factors using the following equation:

\[
\text{ibr}(u,i) = \text{pred}(u,i) + \sum_{v \neq u \in G} (1 - p_u) \cdot \frac{t_{u,v} \cdot (\text{pred}(v,i) - \text{pred}(u,i))}{|G| - 1} + \text{pred}(u,i)
\]

In this equation, \( \text{pred}(u,i) \) is modified according to its difference with the ratings of other users \( (\text{pred}(v,i) - \text{pred}(u,i)) \). This difference is weighted with the trust between users \( (t_{u,v}) \), where \( v \in G \), being \( G \) the group formed by the people who would have the most
influence in user $u$, and therefore, the people who $s/he trusts the most. Finally, the accumulated difference is weighted according to $u$’s personality in an inverse way $(1 - p_u)$.

8) SelectCases: Consists of selecting for each active user $u \in G$ the $k$ items from $T$ whose predicted ratings are highest. For example, in HappyMovie and in HappyShopping, we use $k = 3$. Note that the next 3 tasks are specific for social group recommendations, and therefore the method that implements this task will need to have a display cases option for implementations of just individual recommenders.

9) CombineIndividualRecommendations: Consists of obtaining the group prediction $gpred$ aggregating the predicted ratings of the members of the group, $pred(u, i)$ for each $u \in G$ and $i \in T$ (see Equation 2). Possible aggregation functions ($\bigcup$ in the equation) include least misery (where the minimum is taken) and most pleasure (where the maximum is taken). Methods for aggregating ratings are reviewed in [15]. It is the most pleasure principle that we used in HappyMovie.

$$gpred(G, i) = \bigcup_{u \in G} pred(u, i)$$ (2)

However, in our social group recommendation method [21], [18], we modify the individual ratings with the personality and trust factors. This way, we modify the impact of individual preferences as shown in Equation 3.

$$gpred(G, i) = \bigcup_{u \neq v \in G} f(pred(v, i), p_u, t_{u, v})$$ (3)

where $gpred(G, i)$ is the group rating prediction for a given item $i$; $pred(v, i)$ is the original individual prediction for user $v$ and item $i$; $p_u$ is the personality value for user $u$ and $t_{u, v}$ is the trust value between users $u$ and $v$.

There are several methods to modify the rating predicted for a user according to personality and trust factors. It is represented as the $f()$ function in the Equation 3, some of these ways are the delegation-based method or the influence-based method (see Equation 1) among others. We point interested readers to [21] were several of these social group recommendation methods are detailed.

10) Filtering: Consists of selecting the $k'$ items in $T$ that have the highest predicted ratings for the group. For example, in HappyMovie, we used $k' = 3$.

11) DisplayCases: Consists of displaying to each user $u$ receiving the recommendation the $k'$ items obtained by the group recommender.

12) UpdateGroupHistory: Consists of revising the case $c$ that corresponds to the active user $u$ (for individual recommenders) or the active group of users $G$ (for group recommenders) with the new recommendation and retaining it in the case base $CB$ for future recommendations. Note that this task is optional and can be skipped if developers do not want to build a system with memory of past recommendations.

Figure 3. Relationship between the ARISE architecture, the proposed templates and its instantiation in HappyShopping

IV. HappyShopping

In this section we present HappyShopping$^{3}$: a Facebook social individual recommender application for clothes that follows our generic architecture ARISE and has been built using our social templates. With the development of this application we study and prove the two goals of this paper:

- The usability of our templates (detailed in Section III-A).
- The viability of our generic architecture ARISE (detailed in Section II) in other domains.

In Sections II and III we have detailed how to design a social recommender using ARISE and how to implement it using our social templates, hence, we will only detail now the concrete choices of domain and development that delimit HappyShopping. To understand how each module from the ARISE architecture is defined in one task of the templates and how HappyShopping implements the methods of the needed tasks of the templates, we introduce Figure 3. In the top of the Figure we see ARISE’s modules, each line that goes out of a module points the concrete task in the social templates that corresponds it and each line that comes out of a task in the templates points the concrete module in HappyShopping’s structure that implements it.

$^{3}$http://www.happyspning.es/
A. Details of HappyShopping

Traditional recommender systems do not take into consideration explicit social relations among users, yet the importance of social influence in product marketing has long been recognized [26]. Intuitively, when we want to buy a product that is not familiar, we often consult with our friends who have already had experience with the product, since they are those that we can reach for immediate advice. When friends recommend a product to us, we also tend to accept the recommendation because their inputs are trustworthy. HappyShopping exploits this fact and takes into account preferences of the users’ closest friends in order to recommend which piece of clothing users should purchase and later propose an argumentation process with these closest friends about the recommended items. HappyShopping’s main goal is to present a recommender system that proposes pieces of clothing by taking into account users social context. The recommendation process is summarized in the steps below:

- **Product Comparison with user preferences**: The application requires the user to explicitly identify products that are of her/his interest, which will form the users’ “wardrobe”.
- **Product Comparison with the preferences of most influential friends**: In this step we model the impact of the preferences of the people influencing the user that is being recommended. The proximity between users (users trust) is obtained by analyzing the information available on the social network: messages exchanged, shared photos, etc.
- **Weighting of items regarding the degree of influence of individuals**: The influence of other group members not only depends on their proximity or trust in them, but also in the degree of personality or leadership of these influencers. In this step the products to propose are reconsidered depending on these factors. This process requires obtaining the personality information from the social network.

**Using the HappyShopping system**: Users start their Facebook account and look for HappyShopping in the applications section. HappyShopping’s main page is shown in Figure 4. The required steps to obtain a clothing recommendation with HappyShopping are explained below:

- **Creating the user profile in the application**: Before any user can access the clothing recommendation results users have to create their individual “recommendation profile” which is necessary for our recommendation method. This profile is based on three different aspects: personality, individual preferences and trust in other users.
  - To obtain the personality users have to choose a series of characters to whom they feel identified, Figure 5 up shows HappyShopping’s personality test implementation. This step corresponds to the Personality module in ARISE and its solved by ObtainUsersPersonality task in our template. The concrete method that implements this task is the TKI’s alternative movie metaphor explained in Section III-A.
  - To obtain the preferences profile users have to rate a set of clothes (at least 20 pieces), where they enter their personal preferences, Figure 5 bottom shows HappyShopping’s preferences test implementation. This step corresponds to the Explicit Individual Preferences module in ARISE and its solved by ObtainIndividualPreferences task in our template. The specific pieces that are displayed for the user to rate (users can rate 100 pieces at the most) are selected automatically from HappyShopping’s catalogue trying to maximize the diversity. To do so, a similar metric as the one presented in the system ExpertClerk [25] is used.
  - To obtain the trust, the application reads the information stored in Facebook personal profiles. It calculates the trust that the user has in all the other users in her/his close circle (G). To obtain the circle of trusted people in the social environment of the user receiving the recommendation, the application needs to calculate which other application users should form the group G. This step is solved by the task ObtainGroup in our template. The concrete method that implements this task is the Calculating the group of closest friends in the social network explained in Section III-A. Note that the trust value is general to friends and not specific to the domain (clothes) as it has been proven most efficient in [21], [18].
- **Recommendation**: Once the application has obtained the factors that identify each user receiving a recommendation (personality, individual preferences and trust
in other users) user are able to click the “See recommended clothes” bottom (see Figure 4) and see their individual recommendations. This step corresponds to the Individual Estimation module in ARISE and its solved by the Scoring task in our template. The concrete method that implements this task is the Influence based recommenders explained in Section III-A.

- **Once the recommendation is made:** HappyShopping provides a list of the best 3 pieces of clothing that the recommender has found in the catalogue. For each of them the user will have two options:
  - Purchase the product. Note that this function is not part of the application.
  - Start an argumentation process with the G members. Where the user will ask her/his closest friends which piece of clothing fits her/him best. Note that this option has not been implemented yet as it is not part of the social templates.

HappyShopping counts with a catalogue of 1887 pieces. This catalogue has been obtained by parsing the web searching for different types of clothes and styles to wear. Each item in the catalogue is formed by a picture of the piece of clothing plus the concrete characteristics of the piece like material, colour, style, size, prize, etc.

**B. Evaluation of using our social templates and ARISE**

Regarding the effort and viability of using ARISE and our social templates for the development of HappyShopping, we have counted with 3 developers. The skills and background of HappyShopping’s developers are summarized in Figure 6, that reflects the average of the answers given by the 3 developers to a questionnaire about how they grade themselves. These developers have reused our generic architecture ARISE and its associated templates. We have questioned them about the usability of the set of templates and they all answered that the templates had facilitated and quickened their work thus they all preferred to have the templates to assist them. About the effort that they put on the construction of HappyShopping, we asked them how long it took them to build an initial version of the application, they answered that it took them 5 weeks to develop an initial version and 10 weeks to develop the final version of HappyShopping. If we compare these results with the time that took us (the authors) to develop HappyMovie (which is our other social recommender application in the movies domain as introduced in the previous sections) we can conclude that the usage of our social templates and ARISE has been a success. It took us more than 5 months to develop HappyMovie, and we were 3 expert programmers specialized in CBR and recommender systems. Obviously, as it was the first time that the social recommender system was being implemented there was a high cost in the design and development of HappyMovie, which has been captured in the social templates and the generic architecture and makes the cost of a second social recommender application descend. Therefore, we consider that the use of our social templates and ARISE indeed facilitates and eases the construction of other social recommender applications.

**V. CONCLUSIONS AND FUTURE WORK**

In this paper we have presented a generic architecture ARISE and a set of templates that formalize the behaviour of social recommender systems. We have proven ARISE’s suitability by building two different recommending applications in two different domains. HappyMovie is a particular instantiation of ARISE for recommending movies to groups of people connected through the social network Facebook. The second case study, HappyShopping, is an individual recommender system that follows our method of making
recommendations to people using their social information stored in the social network Facebook. We have also presented a set of templates that represent an intermediate step in the development of social group recommender applications and proven that they quicken and facilitate the process of building new applications. Thus, developers prefer to use these templates than starting a new application from scratch. There is much that can be done to take this work forward. For us, the next step is taking HappyShopping one step forward and make it a richer application with actual ratings from users, from which we hope to gather data and use it as the basis for future experiments. We also want to develop the argumentation process step of the application that was mentioned in Section ??.

ACKNOWLEDGMENT

Supported by Spanish Ministry of Science & Education (TIN2009-13692-C03-03) and Madrid Education Council and UCM (Group 910494) and Spanish Ministry of Economy and Competitiveness under grant IPT-2011-1890-430000.

REFERENCES


