Please cite this article in press as:
Briguez, Cristian E, Budán, Maximiliano CD, Deagustini, Cristhian AD, Maguitman, Ana G, Capobianco, Marcela, Simari, Guillermo.
Argument-based mixed recommenders and their application to movie suggestion.
Expert Systems with Applications (2014), http://dx.doi.org/10.1016/j.eswa.2014.03.046

Argument-based Mixed Recommenders and their Application to Movie Suggestion

Cristian E. Briguez[a,b], Maximiliano C. D. Budán[a,b], Cristhian A. D. Deagustini[a,b], Ana G. Maguitman[a,b], Marcela Capobianco[a,b], Guillermo R. Simari[a]

[a] *Artificial Intelligence Research and Development Laboratory*
Department of Computer Science and Engineering
Universidad Nacional del Sur - Alem 1253, (8000) Bahía Blanca, Buenos Aires
[b] *Consejo Nacional de Investigaciones Científicas y Técnicas*
Av. Rivadavia 1917, Ciudad Autónoma de Buenos Aires - República Argentina

Abstract

Recommender systems have become prevalent in recent years as they help users to access relevant items from the vast universe of possibilities available these days. Most existing research in this area is based purely on quantitative aspects such as indices of popularity or measures of similarity between items or users. This work introduces a novel perspective on movie recommendation that combines a basic quantitative method with a qualitative approach, resulting in a family of mixed character recommender systems. The proposed framework incorporates the use of arguments in favor or against recommendations to determine if a suggestion should be presented or not to a user. In order to accomplish this, *Defeasible Logic Programming* (DeLP) is adopted as the underlying formalism to model facts and rules about the recommendation domain and to compute the argumentation process. This approach has a number features that could be proven useful in recommendation settings. In particular, recommendations can account for several different aspects (*e.g.*, the cast, the genre or the rating of a movie), considering them all together through a dialectical analysis. Moreover, the approach can stem for both content-based or collaborative filtering techniques, or mix them in any arbitrary way. Most importantly, explanations supporting each recommendation can be provided in a way that can be easily understood by the user, by means of the computed arguments. In this work the proposed approach is evaluated obtaining very positive results. This suggests a great opportunity to exploit the benefits of transparent explanations and justifications in recommendations, sometimes unrealized by quantitative methods.

Keywords: Defeasible Argumentation, Recommender Systems, Qualitative vs Quantitative Recommendations

1. Introduction

Recommendation systems are support mechanisms that assist users in their decision-making process while interacting with large or complex information spaces. Most recommender systems are aimed at helping users to deal with the problem of information overload by facilitating access to relevant items [32]. Recommenders attempt to generate a model of the user or user's task and apply diverse heuristics to anticipate what information may be useful. In order to come up with recommendations, conventional recommender systems rely on popularity indices or on similarity measures between users or contents, computed on the basis of methods coming from the social science, the information retrieval or the machine learning communities.

The human-machine interaction observed in existing recommender methods is particularly rigid. Such services do not provide mechanisms for easily reformulating the criteria under which a recommendation will be based. This means that once a particular recommendation technique is adopted and a set of parameters' values are assigned (which are typically established by the recommendation system's configuration), the user does not have the possibility of naturally introducing new preferences for the system to consider or of indicating how to combine various facts and rules for generating recommendations. Decisions about user preferences are mostly based on heuristics, which rely on ranking previous user choices or gathering information from other users with similar interests. In other words, existing recommender systems do not provide an interaction mechanism to deal in a natural and methodical way with users' preferences in complex environments.

Another central weakness in most current methods of recommendation is the lack of transparency. Most on-line recommender systems act like black boxes, not offering the user any insight into the system logic or justification for the recommendations. This is due to the fact that the quantitative methods adopted by most existing recommender systems do not have a clean underlying model that is easily understandable in a final user level. This makes it hard to provide users with a clear explanation of the factors and procedures that led the system to come up with certain recommendations. As has been shown by previous studies (e.g., [43, 45]) users prefer recommendations if they can understand the reasons why these recommendations are presented. Certainly, a recommendation is more compelling if the user is aware of and agrees with the reasons supporting why the recommendation was presented than if it emerges from a black box. In addition, by offering a justification a system is providing additional useful information to its users about related topics or items that they may be unaware of.

Another problem faced by recommender systems is that users' preference criteria generally involve incomplete and potentially inconsistent knowledge about the domain. This is due to the fact that users preferences are dynamic and typically change as time evolves or as new material becomes available for analysis. Because the quantitative approaches adopted by most recommender systems are not equipped with a mechanism to revise previous conclusions, the changing

nature of users preferences is poorly dealt with. Adopting a more qualitative approach by augmenting recommendation with classical logic inference will not solve the above mentioned issues, as it will often lead to contradictory conclusions, which are problematic unless treated appropriately.

In light of these problems, we propose to model users' preference criteria using DeLP [16], a framework for defeasible argumentation [38, 42] based on logic programming that can effectively deal with incomplete and contradictory information. The use of DeLP allows to integrate dialectical reasoning into the recommendation process, which provides a reasoned basis for the items suggested to the user. In addition to offering a natural means to codify data about the recommendation domain, DeLP can deal effectively with the defeasible nature of reasoning patterns that typically arise in recommendation. Finally, an interesting feature of DeLP, that certainly is proven handy in recommendation settings, is that conclusions obtained through this formalism can be easily explained by the argumentative reasoning process. We will circle back to this feature of DeLP latter to show how it can be exploited to obtain natural language explanations of the reasons behind recommendations.

Despite its importance, the use of argumentation-enabled mechanisms in recommendation settings has not received much attention from the recommender systems' community. Nevertheless, in the last decade several advances in the area have been made, mostly from the theoretical point of view. The problem of empowering recommendation with argumentation has been previously addressed in [11]. In that work the focus is set on a general characterization of *argument-based recommender systems*, described as user support tools where recommendations are based on arguments. That work presents a first general approach towards the use of DeLP as the basis of recommender systems. In particular, the system introduced in the present work can be seen as a particular instance of that general approach. An argument-based recommendation system for the music domain was introduced in [6], where a complete framework for achieving recommendations based on defeasible reasoning is presented. A particularly interesting feature of that framework is its use a two-fold argument preference criterion, which enables to easily adjust the behavior of the systems towards different aspects. Nevertheless, the paper only introduces the theoretical aspects of the proposed recommender system, without evaluating its performance. Moreover, although the use of argument-based explanations is mentioned as an advantage of the proposed recommender, there is no complete description of how these explanations can be provided. Finally, further theoretical analyses and simulations aimed at combining quantitative and qualitative aspects to generate argument-based news recommendations were presented in [5]. In that work, both a theoretical investigation and simulations are presented, but there is no empirical analysis of the proposed system performance.

The work presented here can be seen as a confluence and continuation of the above-mentioned works, drawing some key points from them but also presenting new contributions. In particular, we extend previous results by describing a novel framework for the specific domain of movie recommendation. Therefore, we contribute to the research field of recommender systems in different ways.

First of all, we propose a complete set of postulates accounting for both quantitative and qualitative aspects of the movie domain and describe how these postulates can be implemented by means of DeLPrules. Then, we present a careful investigation of the benefits of incorporating defeasible argumentation into the recommendation process. Also, an important difference between our work and the previous work in argument-based recommenders is that our proposal is empirically evaluated using real-world datasets. In order to complete our evaluations, we use an augmented version of the MovieLens dataset [21] and compute classical evaluation metrics to assess the effectiveness of the proposal. Moreover, the family of recommenders presented in this work are the first DeLP-powered real-world systems using a very large scale (10M) dataset, thus setting a precedence on the applicability of massive argument-based applications. Finally, in this work we describe how to obtain complete explanations on the reasons behind recommendations, exploiting the coherent internal structure of arguments to generate a final explanation that is presented to the users as a natural language sentence. Such sentences can account for any type of explanation (*i.e.*, content-based, collaborative-based or preference-based explanations[45]) providing that they can be modeled by rules. To sum up, the main contributions of this work are the following:

- A novel, mixed approach to movie recommendation is presented. The proposed approach considers quantitative and qualitative aspects of the recommendation domain. These aspects are shown to be fully implementable in DeLP, and the benefits of the approach (*e.g.*, easily shifting from one aspect to the other) are discussed.
- An analysis is performed to study how different sets of postulates can be integrated to develop a family of recommenders with different predictive capabilities.
- It is shown how by changing a single feature of the framework (the argument preference criteria) we can go from content-based recommenders to collaborative filtering ones and the other way around, or even mix them in any way we prefer to develop hybrid recommenders.
- Some examples are provided on how recommendations are generated and how explanations can be built based on arguments.
- The proposal is evaluated using an augmented version of the MovieLens dataset [21].

2. Background

2.1. Recommender Systems

Recommender systems are tools that assist users by facilitating access to relevant items, such as music, books, or movies, or to social elements, such as people or groups. These systems adopt mainly two different views to help

predict information needs. The first approach is known as "user modeling" and relies on the use of a profile or model of the users, which can be created by observing users' behavior (e.g., [31]). The second approach is based on "task modeling", where recommendations are based on the context in which the user is immersed (e.g.,[9]). The context may consist of an electronic document the user is editing, web pages the user has recently visited, etc.

Two main techniques have been used to compute recommendations: content-based and collaborative filtering. Content-based recommenders [3, 37], are driven by the premise that user's preferences tend to persist through time. These recommenders frequently use machine-learning techniques to generate a profile of the active user, typically stored as a list of rated items. In order to determine if a new item is a potentially good recommendation, content-based recommender systems rely on similarity measures between the new items and the rated items stored as part of the user model. Recommender systems based on collaborative filtering [17, 2, 41, 40] rely on the assumption that users' preferences are correlated. These systems maintain a pool of users' profiles associated with items that the users rated in the past. For a given active user, collaborative recommender systems find other similar users whose ratings strongly correlate with the current user. New items not rated by the active user can be presented as suggestions if similar users have rated them highly.

A combination of collaborative-filtering and content-based recommendation gives rise to hybrid recommender systems [36, 12, 10, 34]. Hybrid recommenders typically generate a model of the active user by monitoring the user behavior or by analyzing user's declared interests or feedback. The generated user model is usually combined with the user information needs in order to seek for suggestions. In addition, the system maintains a pool of profiles from other users, making possible the application of collaborative filtering to further refine the selected set of recommendations.

Additional dimensions of analysis for recommender systems are the content of the suggestion (e.g., news, URLs, people, articles, text, products), the purpose of the suggestion (e.g., sales or information), the event that triggers the search for suggestions (by user's demand or proactively), and the level of intrusiveness (none, low, moderate or high).

Research in recommender systems has significantly grown in the last years fueled mainly by the proliferation of social networks. For an up to date review of recommendation systems we refer the reader to [4].

2.2. Defeasible Logic Programming

In our work, we will use the formalism of Defeasible Logic Programming (DeLP) [16] as the key to integrate argumentation into recommender systems. The language of DeLP is based on the language of logic programming, where standard logic programming concepts (such as signature, variables, functions, etc.) are defined in the usual way. Literals are atoms that may be preceded by the symbol "$\sim$" denoting *strict* negation, as in extended logic programming. Facts are simply literals. Strict rules are ordered pairs $L_0 \leftarrow L_1, \ldots, L_n$ whose first component, L_0, is a literal, and whose second component, $L_1, \ldots, L_n$, is

a finite non-empty set of literals. Similarly, defeasible rules are ordered pairs $L_0 \prec L_1, \ldots, L_n$ whose first component, L_0, is a literal, and whose second component, $L_1, \ldots, L_n$, is a finite non-empty set of literals. Note that strict rules are used to represent incontrovertible information, whereas defeasible rules are used to represent defeasible knowledge (*i.e.*, tentative information that can be used as long as nothing is posed against it).

In this formalism, the state of the world is modeled as a *Defeasible Logic Program*, essentially a set of facts, strict rules and defeasible rules. In a given defeasible logic program $\mathcal{P}$, the subset of facts and strict rules is referred to as Π, and the subset of defeasible rules as Δ, thus a DeLP program $\mathcal{P}$ can also be noted as (Π, Δ). Since the set Π represents non-defeasible information it must be non-contradictory.

In what follows, we adopt the usual conventions to specify the rules used by our recommenders in the DeLP language. Variable names begin with uppercase letters, while constant and predicate names begin with lowercase letters. For instance, *good_rating(Movie)* $\leftarrow$ *rating_movie(Movie)* $>$ *3* represents a strict rule in this notation (where the constant 3 is a quality bound that represents the low limit value to determine a good quality movie). On the other hand, *recommend(Movie, User)* $\prec$*good_rating(Movie), famous(Actors)* is an example of a defeasible rule.

Given a DeLP program $\mathcal{P}$, a query posed to $\mathcal{P}$ is a ground literal Q which must be supported by an *argument*. Hence, to determine whether a movie should be recommended to a certain user or not, our system must compute arguments based on the rules in the DeLP program. Arguments are built on the basis of a *defeasible derivation* computed by backward chaining applying the usual SLD-inference procedure used in logic programming [25].

Definition 1. *(Argument structure)[16] Let h be a literal, and $\mathcal{P} = (\Pi, \Delta)$ be a DeLP program. We say that $\langle A, h\rangle$ is an* argument structure *for h, if and only if, A is a set of defeasible rules from $\mathcal{P}$ (i.e., $A \subseteq \Delta$), such that: (1) there exists a defeasible derivation for h from $\Pi \cup A$, (2) the set $\Pi \cup A$ is non-contradictory, and (3) A is minimal with respect to set inclusion (i.e., no $A' \subset A$ satisfies the previous conditions).*

Arguments can attack each other. An argument $\langle A_1, h_1\rangle$ *attacks* $\langle A_2, h_2\rangle$ at h if and only if there exists a sub-argument structure $\langle A, h\rangle$ from $\langle A_2, h_2\rangle$ such that $\Pi \cup \{h, h_1\}$ is contradictory. Defeat among arguments is defined combining the attack relation and a preference criterion "$\preceq$". An argument $\langle A_1, h_1\rangle$ *defeats* $\langle A_2, h_2\rangle$ if $\langle A_1, h_1\rangle$ attacks $\langle A_2, h_2\rangle$ at a literal h and $\langle A_1, h_1\rangle \preceq \langle A, h\rangle$ (proper defeater) or $\langle A_1, h_1\rangle$ is unrelated to $\langle A, h\rangle$ (*blocking defeater*).

In order to decide if a partial attack succeeds a comparison criterion must be used, establishing the relative strength of the arguments involved in the attack. This criterion will then solve conflicting situations between arguments. In our recommender system we have chosen to use a particular instantiation of the DeLP preference criteria that combines the following elements:

- *Priorities Among Rules*: a predefined preference order among rules is used to determine which argument prevails. To do this, a partial order relation among rules must be defined. Then, when comparing two arguments, we consider the two rules that have conflicting conclusions, and the winner argument is the one that contains the rule that is preferred.
- *Generalized Specificity* [16, 44]: those arguments that are based on more information or those that support their conclusions more directly are preferred.

Given these criteria, we must describe how they are combined to establish which arguments prevail in an attack situation. Basically, we give the *Priority Among Rules* criterion preponderance over *Generalized Specificity*. In a forthcoming section we will show how this order between the two criteria will enable us to establish a preference relation between different ingredients involved in a recommendation.

Definition 2. *(Defeat) Let $\mathcal{P} = (\Pi, \Delta)$ be a DeLP program. Let $\langle A_1, h_1\rangle$ and $\langle A_2, h_2\rangle$ be two arguments in $\mathcal{P}$. We say that $\langle A_2, h_2\rangle$* defeats *$\langle A_1, h_1\rangle$ if and only if there exists a sub-argument $\langle A, h\rangle$ of $\langle A_1, h_1\rangle$ such that $\langle A_2, h_2\rangle$ counter-argues $\langle A_1, h_1\rangle$ at literal h and it holds that:*

1. *$\langle A_2, h_2\rangle$ is preferred by priority to $\langle A, h\rangle$ (proper defeater), or*
2. *$\langle A, h\rangle$ is not preferred by priority to $\langle A_2, h_2\rangle$ and $\langle A_2, h_2\rangle$ is strictly more specific that $\langle A, h\rangle$ (proper defeater), or*
3. *$\langle A_2, h_2\rangle$ is unrelated to $\langle A, h\rangle$ (blocking defeater).*

Example 1. *Consider the following sets of rules:*

$$\Delta = \{\ recommend(Movie, User) \prec good\_rating(Movie),$$
$$\sim recommend(Movie, User) \prec likes\_by\_top\_genre(Movie, User)\}$$
$$\Pi = \{\ \textit{likes\_by\_top\_genre(matrix, henry), good\_rating(matrix)}\}.$$

Suppose that we establish a priority among rules as follows:

$$\sim recommend(Movie, User) \prec \sim likes\_by\_top\_genre(Movie, User)$$
$$\textit{has higher priority than}$$
$$recommend(Movie, User) \prec good\_rating(Movie)$$

As stated before, this established priority can be used to solve clashes between arguments by transferring such priority to the arguments. Then:

$$A = \{\ \sim recommend(matrix, henry) \prec \sim likes\_by\_top\_genre(matrix, henry)\}$$
$$\textit{is preferred to}$$
$$B = \{recommend(matrix, henry) \prec good\_rating(matrix)\},$$

as the rule in the structure of argument A is preferred to the one in the structure of argument B. Finally, the conclusion is not to recommend the movie "matrix".

In DeLP a literal h will be warranted if there exists a prevailing argument structure $\langle A, h\rangle$. In order to establish whether $\langle A, h\rangle$ prevails, the set of defeaters for A will be considered. Defeaters are arguments and may in turn be defeated. Thus, a complete dialectical analysis is required to determine which arguments are ultimately accepted. Such analysis results in a tree structure called *dialectical tree*, in which arguments are nodes labeled as undefeated (`U-nodes`) or defeated (`D-nodes`) according to a marking procedure.

Definition 3. *(Dialectical tree) [16] The* dialectical tree *for an argument $\langle A, h\rangle$, denoted $\mathcal{T}_{\langle A,h\rangle}$, is recursively defined as follows: (1) A single node labeled with an argument $\langle A, h\rangle$ with no defeaters (proper or blocking) is by itself the dialectical tree for $\langle A, h\rangle$; (2) Let $\langle A_1, h_1\rangle, \langle A_2, h_2\rangle, \ldots, \langle A_n, h_n\rangle$ be all the defeaters (proper or blocking) for $\langle A, h\rangle$. The dialectical tree for $\langle A, h\rangle$, $\mathcal{T}_{\langle A,h\rangle}$, is obtained by labeling the root node with $\langle A, h\rangle$, and making this node the parent of the root nodes for the dialectical trees of $\langle A_1, h_1\rangle, \langle A_2, h_2\rangle, \ldots, \langle A_n, h_n\rangle$.*

For the marking procedure we start by labeling the leaves as `U-nodes`. Then, for any inner node $\langle A_2, Q_2\rangle$, it will be marked as a `U-node` if and only if every child of $\langle A_2, Q_2\rangle$ is marked as a `D-node`. If $\langle A_2, Q_2\rangle$ has at least one child marked as a `U-node` then it is marked as a `D-node`. This marking allows us to characterize the set of literals sanctioned by a given DeLP program, called *warranted literals*. A literal h is *warranted* if and only if there exists an argument structure $\langle A, h\rangle$ for h, such that the root of its dialectical tree $\mathcal{T}_{\langle A,h\rangle}$ is marked as a `U-node`. We will use $\mathcal{T}^{\star}_{\langle A,h\rangle}$ to refer to the marked dialectical tree $\mathcal{T}_{\langle A,h\rangle}$.

Example 2. *In Figure 1 we display the marking process for two dialectical trees. In these examples we represent an argument through a triangle labeled with the argument name and the claim that supports the argument. The arrows represent an attack between arguments. The dialectical tree shown on the left-hand side of the figure contains a non-defeated argument structure that supports the literal h_1. The opposite case is illustrated by the dialectical tree that appears on the right-hand side of the figure.*

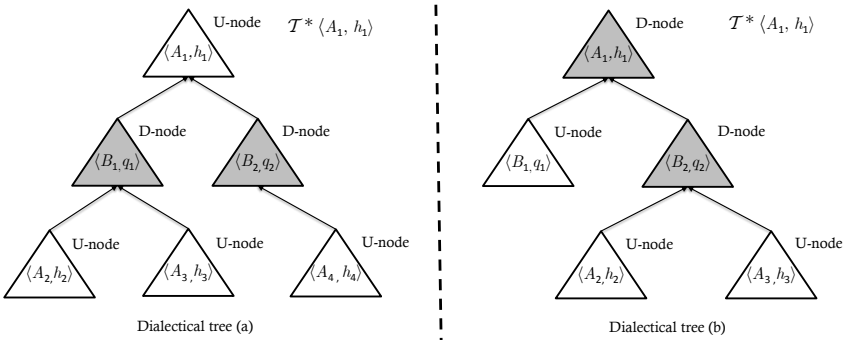


Figure 1: Dialectical tree examples.

In DeLP, in order to determine whether a particular literal is warranted, it is necessary to perform a query on the ground literal that it will try to warrant. A DeLP-query Q is warranted from a DeLP-program $\mathcal{P}$ if there exists an argument A supporting Q such that A is the root of a dialectical tree and the root is marked as undefeated. The answer for a query Q from a DeLP-program $\mathcal{P}$ is either: **YES**, if Q is warranted from $\mathcal{P}$; **NO**, if the complement of Q is warranted from $\mathcal{P}$; **UNDECIDED**, if neither Q nor its complement are warranted from $\mathcal{P}$; or **UNKNOWN**, if Q is not in the language of the program $\mathcal{P}$.

2.3. The DBI-DeLP framework: enabling database supported argumentation

In the previous sections we have described the DeLP formalism and we have stated that it provides an appropriate means for dealing with several issues that arise in the context of recommendation systems. However, to be useful in real-world applications, massive data-driven systems cannot rely on information manually added to a DeLP program. Therefore, the integration of knowledge-based systems (such as recommenders) with large repositories is widely acknowledged as a central issue in the development of real-world scale systems [18, 27]. As an indication of this, consider the efforts made over the years to connect artificial intelligence technologies with large repositories in the form of databases by various approaches in areas such as data mining [15, 26], machine learning [35], or deductive databases [46, 39].

Thus, in order to develop useful argument-based recommenders we have to feed them with live, real-wold information about the recommendation domain and about the users of the system. Instead of including such information directly into the program as facts, we will use relational databases to store this information. In our proposal, the integration of DeLP with a relational database is enabled by the *Database Integration for Defeasible Logic Programming* (DBI-DeLP) framework [13]. The rest of this section briefly introduces the DBI-DeLP framework.

Basically, a DBI-DeLP program is a DeLP program extended with information obtained from databases. In characterizing the framework, it is necessary to consider the possible presence of contradictory information tied to the use of several databases. Given a database D_i, it might be the case that tuples $t_1 \in D_i$ and $t_2 \in D_i$ reflect contradictory knowledge, what makes D_i inconsistent, *i.e.*, information in a dataset can give us reasons both in favor of the recommendation of an item and against it at the same time. This is important for the formalization of the framework underlying our recommenders, because it means that such information cannot be included in the set Π of the program, as it may lead to inconsistency. To overcome this problem, we adopt the notion of *presumption* [16, 33] for representing "defeasible" information, avoiding in that manner inconsistencies not allowed in the strict knowledge available, as required by DeLP. Thus, tuples in databases (in our setting, the information coming from the dataset) are represented as a particular kind of presumptions called operative presumptions, which are literals in the form $pred(q_1, \ldots, q_m) \prec true$.

We extend DeLP programs to include information as operative presumptions obtained from databases. A DBI-DeLP program accounts for a DeLP program

as defined in Section 2.2 along with a set Σ of operative presumptions, associated with the records in the dataset used in a recommender. Such operative presumptions are retrieved on demand by the recommender for solving a particular query associated with a recommendation and are then discarded after the final answer is obtained. Formally:

Definition 4. *(DBI-DeLP Program)*
Let $\boldsymbol{D} = \{D_1, \ldots, D_n\}$ be a set of databases, $\mathcal{P} = (\Pi, \Delta)$ a DeLP program, X the set of all predicates in the rules of $\mathcal{P}$. A DBI-DeLP program $\mathcal{P}'$ is a triplet (Π, Δ, Σ) where $\Sigma = OPset_{\mathbf{X},\mathbf{D}}$ is the set of operative presumptions for $(X, \boldsymbol{D})$.

In order to illustrate these concepts, consider an Argument-based Movie Recommender System containing strict rules such as:

$$child\_restricted(Movie) \Longleftarrow has\_violence(Movie),$$

and defeasible rules such as

$$has\_violence(Movie) \prec director(Movie, tarantino).$$

Dynamically, operative presumptions such as

$$film\_genre(pirates\_of\_the\_caribbean, comedy) \prec true, \text{ and}$$

$$film\_genre(pirates\_of\_the\_caribbean, action) \prec true$$

could be added if different genres for the movie Pirates of the Caribbean are found in a dataset when attempting to solve a query.

In a nutshell, to obtain relevant data for the argumentation process we use the elements in the literal that the dialectical procedure is trying to warrant to determine which records from the database are related to it, then we use the pertinent SQL queries. Finally, we transform all the retrieved results into operative presumptions in such a way that they become available to the argumentation process. This dynamic search for relevant information is crucial if we want to obtain relevant data (*e.g.*, the movies a certain user has seen) from the universe of available data in the dataset.

Due to space constrains, no further details of DBI-DeLP are presented here. The interested reader is referred to [13] for a more complete description of the framework.

Further in the paper we will show a complete example of how recommendations using real-world data stored in a relational database can be accomplished, and how the information retrieval process of DBI-DeLP is a cornerstone of it.

3. An argument-based movie recommender

As it has been pointed out earlier [11], recommendation approaches based on purely quantitative criteria often fail at giving users the reasons behind recommendations, thus affecting the users' trust in the results. In qualitative approaches such as the one we propose in this paper, explanations naturally

complement recommendations. Therefore, the user will receive both a recommendation and a reason supporting it. This has a two-fold advantage: the user will have more confidence on the presented result, and the user can give the system explicit feedback that can help guide the recommendation process.

In the previous sections we have introduced the underlying theory for developing an argument-based recommender. In the present section we show how both quantitative and qualitative criteria can be modeled using DeLP rules to implement a mixed-character recommendation system. This mixed approach to recommendation allows to generate reasoned recommendations based on massive real-world datasets.

3.1. Dataset

The data we have used for our tests is based on a dataset from Grouplens [22], known as the 10M MovieLens dataset. The selected dataset contains 10,000,054 ratings. The ratings are on a scale from 1 to 5 stars. The dataset also contains 95,580 tags assigned by 71,567 users to 10,681 movies of the online movie recommender service MovieLens [21]. To complete our tests, users were selected at random. All the selected users had rated at least 20 movies. Unlike others MovieLens datasets, no demographic information was available. Each user is represented by an id, and no other information is provided.

Figure 2 illustrates the structure of the three files containing the data about movies, ratings and tags.

Movies	Ratings	Tags
ID	Movie ID	User ID
Title (Year)	User ID	Movie ID
Genre List	Rating	Tag
	Timestamp	Timestamp

Figure 2: Data files structure.

Among the datasets provided by GroupLens we selected the largest one with the purpose of showing that DeLP is capable of dealing with massive amounts of data. Since we wanted to base our recommendations on certain attributes that were not available in these datasets, we have augmented them by taking the Internet Movie Database (IMDb) [24] as a source to add data such as directors, lead actors, writers, etc. These additional attributes constituted an important ingredient to study the impact of qualitative aspects on the recommender performance. Figure 3 illustrates the resulting set of attributes used to characterize a movie.

We have noticed that Grouplens has also developed a dataset with features similar to ours [20]. However, the dataset we have created is more than ten times larger than the one provided by Grouplens. The dataset used in this work can be downloaded from **http://ir.cs.uns.edu.ar/datasets**.

Movie Data
ID
Title
Year
Release Date
IMDB URL
Rated
Runtime
Rating
Rating Count IMDB
IMDB ID
Actor List
Director List
Writer List
Genre List

Figure 3: Stored data for each movie.

3.2. Supporting recommendations: quantitative and qualitative aspects

The mechanism for generating recommendations in an argument-based recommender system is very different from the process adopted by most of the existing recommenders. Nevertheless, they share the same spirit: they attempt to establish the similarity between objects or users, and then they use this similarity to make recommendations. The main difference is that while purely quantitative approaches compute this similarity as a numerical measure, in the argument-based approach proposed here the similarity is determined using rules that state which characteristics have to be shared between objects or users to be considered similar.

The proposed recommendation approach relies on a set of intuitive postulates that we have developed to describe the conditions under which a movie should be recommended to a given user. These postulates are presented in Table 1:

Postulate		Description
1	1.1	A user may like a movie if the average movie´s rating is above the average general rating.
	1.2	A user may dislike a movie if the average movie´s rating is under the average general rating.
2	2.1	A user may like a movie if the genre of the movie is one of the user´s favorite ones.
	2.2	A user may dislike a movie if it has one of the genres that are most disliked by the user.
3	3.1	A user may like a movie if the actor of the movie is one of the user´s favorite ones.
	3.2	A user may dislike a movie if it has one of the actors that are most disliked by the user.
4	4.1	A user may like a movie if he likes another movie that has at least one actor in common and has the same genre.
	4.2	A user may dislike a movie if he dislikes another movie that has at least one actor in common and has the same genre.

Table 1: Postulates.

These postulates can be easily translated into DeLP rules. The applicability of the rules will depend on the information in the dataset. For the sake of

simplicity, from this point on we will consider that certain literals are built-in. These literals are computed elsewhere, retrieving information from the dataset (by means of the approach presented in Section 2.3) and using it to prove the literals under analysis. For instance, the system uses the literal *genre(Movie, Genre)* to retrieve information from the dataset about the genres of a certain movie.

Postulate		Description
1	1.1	good_movie (Movie) $\prec$ avg_rating(Movie) > 3.8.
	1.2	~ good_movie (Movie) $\prec$ avg_rating(Movie) $\leq$ 3.8.
2	2.1	likes_by_genre (Movie, User) $\prec$ top_genre(User, Genre), genre(Movie, Genre).
	2.2	~ likes_by_genre (Movie, User) $\prec$ bottom_genre(User, Genre), genre(Movie, Genre).
3	3.1	likes_by_actor (Movie, User) $\prec$ top_actor(User, Actor), leads_in(Movie, Actor).
	3.2	~ likes_by_actor (Movie, User) $\prec$ bottom_actor(User, Actor), leads_in(Movie, Actor).
4	4.1	likes_by_actor_genre (Movie, User) $\prec$ top_actor(User, Actor), leads_in(Movie, Actor), top_genre(User, Genre), genre(Movie, Genre).
	4.2	~ likes_by_actor_genre (Movie, User) $\prec$ bottom_actor(User, Actor), leads_in(Movie, Actor), bottom_genre(User, Genre), genre(Movie, Genre).

Table 2: Postulates - DeLP Rules.

Notice that these postulates use auxiliary predicates to find out the favorite/most disliked actors and genres for a certain user. These predicates are computed as follows:

- top_actor: we have two ways to determine the favorite actors. The first one is to obtain the three actors that appear most frequently in the set of movies watched by the user. An alternative way is to consider only those movies that the user has rated with a four or greater value.
- bottom_actor: is obtained as the three actors that appear most frequently in the set of movies that the user has rated with a two or lower value.
- top_genre: is defined as the three genres that appear most frequently in the set of movies that the user has rated with a four or greater value.
- bottom_genre: is obtained as the three genres that appear most frequently in the set of movies that the user has rated with a two or lower value.

Also, to use the postulates we need to encode an auxiliary set of rules in the form *recommend(Movie, User)* $\multimap$ *likes_by_postulate(Movie, User)*. So a particular instantiation of the rule will be, for example, *recommend(Movie, User)* $\multimap$ *likes_by_actor(Movie, User)*.

The value 3.8 used in some of the postulates is the result of averaging all the ratings available in our dataset. Therefore, one of the proposed postulates states that a movie is good if its average rating is above the average of all the ratings at hand. On the other hand, a movie is not good if its average rating is less than or equal to that value.

The current system implementation is based on the set of postulates described in Table 1, which as can be seen correspond to a content-based approach. Nevertheless, this is a particular feature of this version of the recommender, and does not represent a restriction of the underlying framework. We can easily expand the recommender by incorporating additional postulates accounting for collaborative-filtering aspects. Table 3 describes a set of extra postulates exploiting the relation among similar users that could be used to extend the basic postulates, in order to transform the current implementation into a hybrid one.

Postulate		Description
5	5.1	A user may like a movie if the movie is liked by a similar user.
	5.2	A user may dislike a movie if the movie is disliked by a similar user.
6	6.1	A user may like a movie if the movie is liked by a group of similar users.
	6.2	A user may dislike a movie if the movie is disliked by a group of similar users.

Table 3: Postulates - Extra.

3.3. Deciding what is important: relevant aspects in the final recommendation

Based on the given set of postulates it is possible to obtain arguments in favor or against the potential recommendations. A problem arising from this situation is how to decide whether to finally recommend a movie or not. We have introduced an argument preference criterion which combines two different criteria that will help to solve this problem. We will use the priority among rules preference criterion to state which postulates prevail over others. This gives us the opportunity to easily find the better combination of postulates by empirical trials. So, we can go from a collaborative-filtering-based recommender which refines its answers on content-based aspects to the other way around effortlessly. We can even go a step further and mix aspects in any arbitrary way we want, just by switching priorities between rules.

The second preference criterion, generalized specificity, is used just for the sake of simplicity, as it will solve conflicts in those cases where we do not want to establish explicit priorities. In particular, this is useful when a rule is used to refine the conditions stated in another one. For example, according with this criterion, given the rules *recommend(Movie, User)* $\prec\!\!\!-$ *likes_by_postulate(Movie, User)* and $\sim$*recommend(Movie, User)* $\prec\!\!\!-$ *likes_by_postulate(Movie, User), is_counter-example(Movie, User)*, those arguments built with the second rule will be more specific than those built with the first rule.

3.4. Expanding aspects: addition of rules

We have already shown how different aspects that can be taken into account for recommendations can be included in an argument-based recommender system by modeling them through combinations of strict and defeasible rules. Moreover, as we have mentioned earlier, using an argumentation approach to recommendation like the one introduced before we can mix up the modeled aspect in virtually any way we want. This allows us to build mixed recommender

systems, and to easily adjust them towards being more quantitative or qualitative oriented to better fit the constraints of the particular context in which they will be used. Such characteristics can be certainly proved handy in the development of recommender systems for several particular domains, making our argumentation-based approach a viable option towards such developments.

Another important benefit of using rules to codify the recommendation criteria is that rules represent the cases they model in a colloquial way that is easy to understand and discuss by final users. This characteristic facilitates the interaction between the user and the system; we think it is easier for the final user to give feedback to the system based on natural language explanations to refine rules than to propose the change or addition of a parameter in a mathematical formula.

Finally, a useful characteristic of certain quantitative approaches is that modifications can be made to them in order to refine previous recommendations. This is something that also holds in our argumentation-based recommendation scenario, as the colloquial way in which rules are represented leads to another feature: new cases can be modeled by adding rules to the system at any time, without affecting other rules in the system. That is, we can add rules to model new recommendation aspects directly by encoding them as new strict or defeasible rules. These rules will be considered in subsequent recommendations, possibly modifying the universe of recommended items for certain users.

In this way we can expand the set of postulates to change the criteria upon which an argument-based recommender system generates its suggestions. We argue that in any dynamic enough environment (as is typically the case in all recommendation domains), evolving the way recommendations are made is not only a benefit, but can also be crucial to the system's success. Basically, there are two situations when we may want to expand the aspects that the recommender takes into account. On the one hand, we may want to add a rule that refines an existing one. For instance, a refinement is worthwhile if we discover that a certain rule works well in most cases but it has a number of counter-examples, and we know the characteristics that those counter-examples share. On the other hand, we can add an entirely new postulate modeling a relation between data that we have not been taking into account, or if we suddenly had access to data that was previously absent, as can be the case if a new dataset related to our application domain is developed.

Regarding the first case, a noteworthy advantage of using these high level rules to define recommendations is that they allow to include heterogeneous contextual information, which is typically hard to model in other approaches. Although some quantitative approaches have already addressed this issue [28], the way contextual characteristics are modeled in these approaches (using categorical values) suggests that a qualitative approach may lead to a more natural way to deal with these characteristics in the recommendation process.

In our approach, if we have access to contextual information, we can model it as predicates and include them in the postulates to refine them. In this way, modeling and using different contextual aspects (*e.g.*, weather, time or user mood) at the same time becomes easier, as it is not necessary to define similarity

metrics for these attributes. Instead, important aspects could be refined using more detailed rules. We illustrate this with an example, adapted from [6].

Example 3. *Consider the following rule:*

recommend(Movie, User) $\prec\!\!-$ *likes(User, Movie2), same_genre(Movie, Movie2).*

Thus, we recommend a movie to a user based on the fact that she had liked similar movies in the past. However, suppose that we know that this is typically true except when the movie's genre is action, the user is sad and the weather is rainy. Then, we can refine the previous rule as follows:

$\sim$*recommend(Movie, User)* $\prec\!\!-$ *likes(User, Movie2), same_genre(Movie, Movie2), genre(Movie, "Action"), mood(User, "Sad"), weather("Rainy").*

*This allows the recommender to modify its recommendations based on the information it has about the weather (by connecting to one of the several web services that provide that information) and the current user's mood (*e.g.*, using mood-inference services such as* MoodScope *[30]).*

The addition of rules in argument-based recommenders also facilitate the incorporation of new dimensions of analysis. For instance, we can think that users' preferences slightly change over time, or are at least affected by it. If that is the case, time can be used as a contextual feature in order to generate better recommendations. To accomplish this we can add new rules that modify the recommendations over time, using defeasible argumentation formalisms that take time into account [8].

However, the refinement of rules can be used for more than merely including contextual information. If users' feedback is available, we could look further into those cases where the application of a postulate results in false positives or false negatives. Then, a refinement of the postulate may be formulated in order to help overcome the problem, as in the following situation.

Example 4. *Consider the following rule:*

recommend(Movie, User) $\prec\!\!-$ *genre(Movie, Genre), favorite_genre(User, Genre).*

Then, we recommend a movie to a given user based on the fact that the user likes its genre. However, assume that we find out (through users' feedback) that recommending Hollywood drama movies to users that live in South America often causes false positives, as those users tend to dislike them. Then, a new rule can be added to the program to state this, as follows.

$\sim$*recommend(Movie, User)* $\prec\!\!-$ *genre(Movie, "drama"), favorite_genre(User, "drama"), origin(Movie, "Hollywood"), lives_in(User, "'South America').*

Notice that, as explained before, the interaction among the rules is at the dialectical level, and thus they can coexist because the potential conflicts will be handled by the argumentation process. Clearly this has a direct impact on the recommender's flexibility, as we can further refine recommendations for some subset of users (or items) leaving the rest unchanged. In Example 4 we refine recommendations for South American users, without losing the more general case for users that live in other parts of the world.

4. The role of argumentation in recommendation

As we will show through this section, the use of argumentation allows to give preponderance to those attributes that better define a relevant recommendation. Argumentation also makes it possible to offer explanations as coherent sets of reasons in favor or against a recommendation.

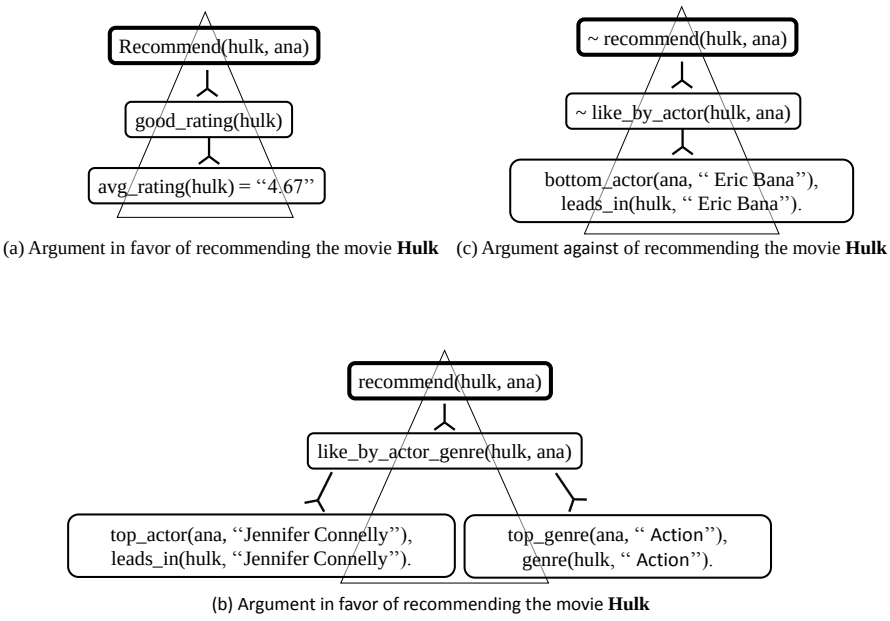
4.1. Argumentation-based recommendations

This section describes how the arguments are constructed, how the dialectical process is carried out, and how it is used to make recommendations to users. As it has been discussed in previous sections, rules in a DeLP program are combined to support or reject a conclusion by building arguments. An argument can be in favor or against a given conclusion, but cannot play both roles at the same time. By contrast with other argumentation systems, arguments in DeLP are derived from the logic program under analysis, and they have internal structure. This structure is what allows to provide an explanation on the acceptability or not of a particular conclusion.

Figure 4 shows the arguments that have been computed to make a recommendation to the user *Ana*. In this example we can see how arguments can be built in favor or against recommending the movie *Hulk* to *Ana*. Figure 5 presents a dialectical tree resulting from these arguments.

In section 2.2, we stated that every argument attacked by at least one undefeated argument becomes defeated, and every argument without (undefeated) attackers becomes undefeated. If the root argument of a dialectical tree is undefeated, the dialectical tree represents an argumentation process that supports the conclusion of such argument. In addition, the DeLP reasoner builds a set of dialectical trees, trying to give support or contradicting the arguments for or against the query, and these arguments represent an explanation for such query.

Assume that the query under analysis is ***recommend(hulk, ana)***. In addition, suppose that the recommender system gives preponderance to the movies performed by actors that the user likes, and to movies whose genres are liked by the user over those movies with good rating. In this case, the system will evaluate the actors who interpret the movie *Hulk* (*Eric Bana, Jennifer Connelly and Sam Elliott*) as well as the movie's genres (*Action, Sci-Fi*) to determine if *Ana* likes any of the actors or any of the genres. Since "*Jennifer Connelly*" is among the actors preferred by *Ana* and *Ana* likes "*Action*" movies, this evaluation will be positive. So, we have a reason to believe that the user *Ana* would like the



(a) Argument in favor of recommending the movie **Hulk** (c) Argument against of recommending the movie **Hulk**

(b) Argument in favor of recommending the movie **Hulk**

Figure 4: Arguments in favor and against recommending a particular movie.

movie *Hulk*, and according to our system this reason is better than the reason to believe otherwise, *i.e.*, we give that rule priority over the others. Thus, we finally recommend the movie to the user.

4.2. Offering explanations for recommendations

In our previous examples we have stated the reason behind a recommendation by using colloquial language to describe the structure of the argument that has prevailed in the dialectical analysis. The simplicity of putting down the structure of the prevailing arguments in natural language sentences leads to another advantage of the argumentative approach: giving explanations to recommendations is almost straightforward when they are based on arguments, as they can be seen as self-explanatory set of reasons.

As a consequence, implementing a module for constructing explanations in an argument-based recommender is a straightforward task. All that is needed is to maintain an association between the structure of arguments and the schemas of explanations, where the latter are parametrized by the variables in the structure of the former. This is fairly easy to do, since the number of rules we have is finite (and often small). For the postulates used in this work, the association can be constructed as shown in Table 4. For the sake of simplicity we only introduce how to obtain explanations for positive recommendations, but negative ones can be defined in a similar way using the proper postulates.

Then, when we obtain a dialectical tree supporting a recommendation, we can use the supporting argument in a leaf node to obtain the proper values

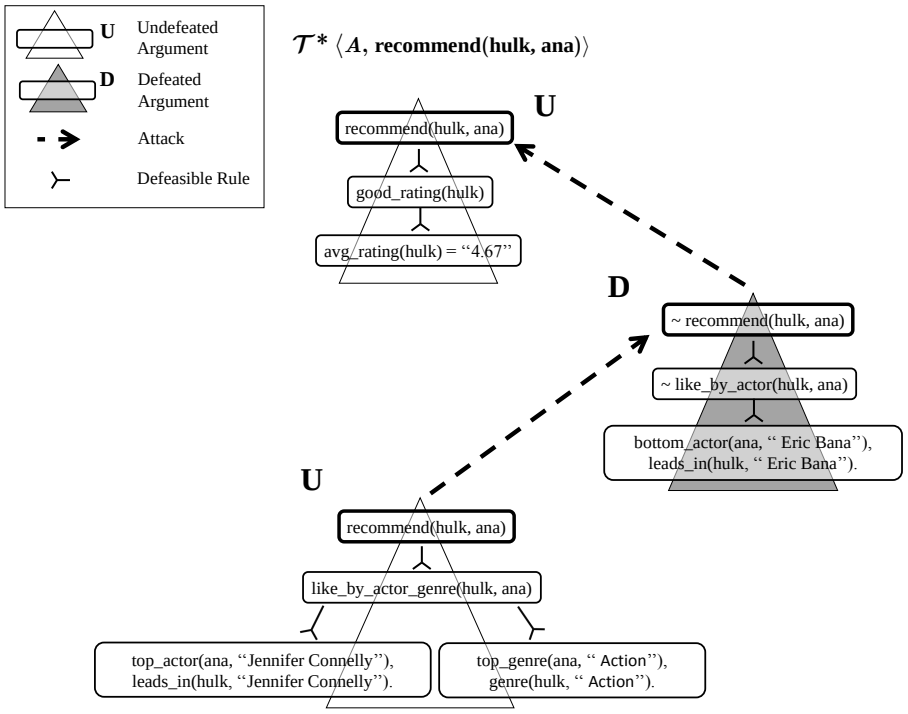


Figure 5: Dialectical tree for recommending the movie Hulk to Ana.

for the variables in the schema of the explanation for the recommendation, thus obtaining a complete explanation that can be presented to the user. For instance, continuing with the recommendation scenario introduced in Section 4.1, we use the argument in the leaf node in the dialectical tree shown in Figure 5 to justify why the movie Hulk is recommended to the user Ana. This explanation will take the form **"Ana, Hulk is recommended to you because it has one of your favorite actors, Jennifer Conelly, in a lead role and it corresponds to one of your favorite genres, Action."**. Offering this kind of explanations could help in building users' trust, as has been studied in [45]. Moreover, it is possible to refine explanations further, by showing Ana the set of movies she has liked in which Jennifer Conelly is a protagonist, as the reason behind considering such actress as one of Ana's favorite ones.

5. Empirical evaluation of the approaches: results and findings

This section evaluates the proposed framework and discusses the findings resulting from these evaluations. The goal of the evaluation presented in this section is two-fold. On the one hand, we carry out a comparison on the performance of the postulates proposed in Section 3.2, in order to find out which

Postulate/Argument Structure	Schema for the reason behind Recommendation
Postulate 1: < recommend(Movie, User), good_movie(Movie) $\prec$ avg_rating(Movie) > 3.8 >	User, Movie is recommended to you because its average rating is above the overall average.
Postulate 2: < recommend(Movie, User), likes_by_top genre(Movie, User) $\prec$ top_genre(User, Genre), genre(Movie, Genre) >	User, Movie is recommended to you because it corresponds to one of your favorite genres: Genre.
Postulate 3: < recommend(Movie, User), likes_by_top actor(Movie, User) $\prec$ top_actor(User, Actor), leads_in(Movie, Actor) >	User, Movie is recommended to you because one of your favorite actors, Actor, has a lead role in it.
Postulate 4: < recommend(Movie, User), likes_by_top actor(Movie, User) $\prec$ top_actor(User, Actor), leads_in(Movie, Actor), top_genre(User, Genre), genre(Movie, Genre) >	User, Movie is recommended to you because it has one of your favorite actors, Actor, in the lead role and it corresponds to one of your favorite genres, Genre.

Table 4: Explanations for Postulates.

ones are the most suitable for developing the proposed argument-based movie recommender system. On the other hand, we compare the performance of a basic quantitative recommender with the performance of a mixed recommender that refines quantitative recommendations with qualitative aspects.

Regardless of the two-fold goal of the evaluation, the experiments completed to asses the performance of the recommenders involve the same actions: we take a movie and attempt to predict whether the user liked the movie or not. To conduct this evaluation, we completed a number of tests, which are described next.

5.1. Experimental setup

The test set used for our evaluations was obtained from the enhanced MovieLens Data Set presented earlier and consisted of 3000 ratings. These ratings were randomly selected with the only provision being that they all came from different users. By ensuring that the ratings used for the test set were associated with different users, we were able to evaluate the postulates for the general case, avoiding the possible introduction of bias coming from the behavior of particular users. On the other hand, the same movie may appear more than once in the test set, as the same movie may occur in two ratings coming from two different users.

The goal of our evaluation was to determine if the system was able to predict if the user liked the movie or not. Therefore, for each rating in our test set we posed the query "recommend(Movie, User)?" to the system to obtain the system prediction. To ensure that the (movie, user, rating) triplet being validated had no influence on the prediction made, we removed the corresponding record from the training set.

An important remark about the possible results given by an argument-based recommender system is that the possible outcomes can be positive (i.e., the movie is recommended to the user), negative (i.e., the movie is not recommended to the user) or undecided. The last case arises when the system cannot warrant recommending or not recommending a movie to a user, leading to an

undecided result as explained in Section 2.2. We would like to stress out that such undecided results should not be interpreted as the recommender saying that the movie X is not recommended to user Y. That is, undecided results are not a particular kind of negative recommendations. Instead, they mean that the recommender neither does support nor does it negate the recommendation. Further in the paper we will have a better look into how undecided results are considered to measure the efficiency of recommendations.

To perform the tests, we pose each of the 3000 queries to the recommender. Then, we classify every response in one of the following classes, according to the response obtained and the rating the user gave to the movie:

- True Positive (TP): the system suggests that the user will like the movie, and the user rated the movie with a four or greater value.
- True Negative (TN): the system suggests that the user will dislike the movie, and the user rated the movie with a three or lower value.
- False Positive (FP): the system suggests that the user should watch the movie, and the user rated the movie with a three or lower value.
- False Negative (FN): the system suggests that the user should not watch the movie, and the user rated the movie with a four or greater value.
- Undecided (U): the system neither suggests nor negates the recommendation.

This experiment is repeated for different subsets of postulates, leading to a comparison among the postulates. This comparison will be discussed in detail later.

5.2. Performance metrics

In order to evaluate different aspects of the resulting family of recommenders we have used several Information Retrieval (IR) performance metrics. While these metrics are common in IR setting, some of them are not entirely suitable for our scenario, where we have undecided results. Notice that undecided results are not exactly a class on its own because movie recommendations are not *intentionally* classified as undecided, but rather they are left unsolved by the system. Then, we are not in the presence of a multi-class classification problem, but we are not dealing with a *purely* binary classification either. For instance, we cannot directly take the obtained results into a standard two-by-two confusion matrix. Rather, we have a binary classification augmented with the additional undecided state. Next we present the performance metrics used in the evaluation of the family of recommender systems, stating whether or not these metrics may be influenced by the presence of undecided results, and how the problem is solved in case they do.

Precision. This is a traditional metric that measures how accurate are the recommendations given by the system. Precision is defined as the number of retrieved relevant resources divided by the total number of retrieved resources. Often, the precision of a system is seen as a measure of the quality of the retrieved resources. In our movie recommendation scenario precision is computed as the number of movies that the system correctly predicts as movies that the user will like to watch (*i.e.*, the number of true positives) divided by the total number of movies recommended positively (*i.e.*, the sum of true positives and false positives). Summarizing, we have that:

$$Precision = \frac{TP}{TP + FP}.$$

Notice that undecided results do not affect the recommender's precision, which makes sense since those movies for which the answer is undecided are not retrieved at all.

Recall. While precision is about quality, recall involves the notion of quantity and measures how exhaustive the recommendations made by a system are. It is defined as the number of retrieved relevant resources divided by the number of relevant resources. In the movie recommendation domain it can be computed by dividing the number of movies correctly recommended by the total number of movies that are worth recommending (regardless whether the prediction is positive, negative or undecided). Thus, recall should be influenced by undecided results as well as positive and negative ones, but we cannot state if an undecided answer stands for a recommendation for the movie or not. For this matter, we have chosen to use the following formula, which directly considers the number of relevant movies in the dataset:

$$Recall = \frac{TP}{RM},$$

where RM is the number of cases (out of the 3000 tested ratings) for which the actual rating given by the user is equal or greater than four. This means that RM accounts for the total number of triplets (movie, user, rating) in our test set where rating ≥ 4 (which means that the movie should be recommended, and hence it is relevant). Notice that such measure may be unusable in some recommendation settings where the number of relevant documents cannot be obtained.

Accuracy. Accuracy stands for the fraction of resources predicted as positive or negative for which the prediction was correct. From this it follows that undecided answers should not influence accuracy because for these cases no prediction is made. Then, we use the usual formula for accuracy, which is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

F-Score. A measure that considers both the quality and the quantity of the set of resources retrieved by a system at the same time is the F-score, which is the weighted harmonic mean of precision and recall. The general formula for the F-score weighted by a parameter β, which we denote by $F_{(\beta)}$, is:

$$F_\beta - score = \frac{(1+\beta^2) \times (precision \times recall)}{(\beta^2 \times precision + recall)},$$

where β is a non-negative real number such that if $\beta = 1$ then the formula measures recall and precision with the same weight, if $\beta < 1$ then precision is considered more important, and otherwise recall is more important.

5.3. Results

Next we present the results obtained in the experiments that were carried out to test the different recommendation approaches. The first set of results shed light on the effectiveness of different combinations of postulates. Then, the second set of results is used to determine if a mixed recommendation approach accounting for both quantitative and qualitative aspects performs better than a purely quantitative approach.

5.3.1. Recommendation effectiveness of the different postulates

In order to compare the effectiveness of the different sets of postulates, we have calculated the metrics proposed in Section 5.2 for each one of the postulates as well as for different combinations of them. As a remark, the number of records with relevant movies (i.e., those records out of the 3000 records from the test set where the user had rated the movie with a value equal or greater than four) is 1977. Table 5 presents the results for different sets of postulates using the test set consisting of 3000 ratings. The performance metrics computed for the different combinations of postulates is summarized in Table 6.

Postulates \ Type of answer	TP	TN	FP	FN	UN
Postulate 1	1449	609	414	528	0
All Postulates	1619	585	441	355	0
Postulate 2	551	163	142	132	2012
Postulate 3	576	97	57	21	2249
Postulate 4	129	17	3	2	2849
Postulate 2 + 3	991	234	194	117	1464
Postulate 2 + 4	551	163	142	132	2012
Postulate 3 + 4	529	85	47	14	2325
Postulate 2 + 3 + 4	991	234	194	117	1464

Table 5: Results for different sets of postulates.

Notice that the lowest precision (0.78) is achieved when only Postulate 1 is used. This case can be taken as a baseline for comparison, as Postulate 1 takes into account quantitative information only. On the other hand, the best

Postulates \ Metric	Precision	Recall	Accuracy	$F_{(1)}$	$F_{(0.5)}$
Postulate 1	0.78	0.73	0.41	0.75	0.77
All Postulates	0.79	0.82	0.44	0.80	0.79
Postulate 2	0.80	0.28	0.24	0.41	0.58
Postulate 3	0.91	0.29	0.25	0.44	0.64
Postulate 4	0.98	0.07	0.07	0.12	0.26
Postulate 2 + 3	0.84	0.50	0.35	0.63	0.58
Postulate 2 + 4	0.80	0.28	0.24	0.41	0.58
Postulate 3 + 4	0.92	0.27	0.23	0.41	0.62
Postulate 2 + 3 + 4	0.84	0.50	0.35	0.63	0.58

Table 6: A summary of the performance of the different sets of postulates.

precision value (0.98) was achieved when Postulate 4 is used. This postulate combines qualitative information about the actors and about the genres that the user likes. Nevertheless, it is clear that such a high precision, achieved by Postulate 4, comes at the cost of a very poor recall (0.07), which can be interpreted as that the recommender based on this postulate only recommends a movie to a user if it has high confidence that the user will like it.

Also, it is interesting to note that the qualitative postulates (Postulates 2, 3 and 4) are superior in terms of precision to the combination of qualitative and quantitative postulates (All Postulates). Nevertheless, we should observe that the quantitative approach achieves considerably superior recall than the qualitative approaches. While for the movie recommendation domain, precision is more important than recall, other application domains, such as medical diagnosis, may greatly benefit from achieving high recall. As a final remark on this matter, we would like to stress out that for the case when *all* postulates are considered together the number of cases solved by qualitative recommendations for the experiments presented in the paper is the sum of the cases solved by considering Postulates 2, 3, and 4 altogether; that is, the sum of the cases in the final row in Table 5, which adds up to 1536. Thus, when we considered together the quantitative and qualitative postulates in a mixed recommender more than 50% of the cases (1536 out of 3000) are solved by taking into account aspects arising from the qualitative approach.

5.3.2. Quantitative approach vs. mixed approach

The specific goal of this part of the experimentation is to shed some light on the question of whether augmenting a basic quantitative recommender with qualitative aspects does in fact result in significant improvements.

In order to measure this, we have posed the 3000 queries to two different DBI-DeLP programs. The first program has as its only case for recommendation Postulate 1, and hence it is a purely quantitative recommender using only the average rating of movies for recommendations. The second program stands for the mixed approach and uses Postulates 1, 2, 3 and 4 to determine if a certain movie should be recommended to some user or not. As a remark, for the mixed

recommender the priority among postulates is as follows:

$$Postulate\,4 > Postulate\,3 > Postulate\,2 > Postulate\,1.$$

The reason behind this choice is that, for our movie recommendation scenario, we prioritize precision over recall, as we prefer to recommend a small number of movies to the users if that means that the users will tend to agree with our recommendations. Then, we have chosen to establish this priority according to the results shown in Section 5.3.1. Notice that the presented priority scheme means that the most important Postulates for this particular recommender are the qualitative ones. Thus the disagreement between the quantitative approach and some of the qualitative approaches is always solved in favor of the qualitative ones. In practice, this causes that answers obtained based on the quantitative aspects are refined (by overriding them) by the qualitative aspects, even for those cases when the recommendations coincide.

The results of comparing the purely quantitative approach with the mixed approach are summarized in Table 7:

Postulates \ Metric	TP	TN	FP	FN
Quantitative recommender	1449	609	414	528
Mixed recommender	1619	585	441	365

Table 7: Results obtained by the quantitative and mixed approaches.

In order to determine if the improvements achieved by the mixed recommenders were statistically significant, we divided the 3000 ratings in our test set into 30 samples of 100 items each. Based on the obtained results, a statistical analysis was performed. Table 8 presents the results obtained by the quantitative approach, and Table 9 presents the ones for the mixed approach. We report the means (MEAN), standard deviations (STDEV) and confidence intervals at the 95% level (95% C.I.) for each type of answer.

Type of answer \ Parameter	MEAN	STDEV	95% C.I.
TP	48.30	4.92	[46.54, 50.06]
TN	20.30	4.04	[18.86, 21.74]
FP	13.80	2.68	[12.84, 14.76]
FN	17.60	3.73	[16.27, 18.93]
TP + TN	68.60	4.38	[67.03, 70.17]

Table 8: The mean, standard deviation and 95% confidence interval for the quantitative approach.

5.4. Analysis of findings

Several conclusions can be drawn from the previous experiments considering first which is the best combination of Postulates in our movie recommendation scenario and second whether or not the refinement of a basic quantitative approach with qualitative aspects does in fact improve performance. We

Type of answer \ Parameter	MEAN	STDEV	95% C.I.
TP	53.97	5.16	[52.14, 55.80]
TN	19.50	4.37	[17.94, 21.06]
FP	14.70	3.17	[13.56, 15.83]
FN	12.17	3.73	[10.83, 13.50]
TP + TN	73.47	3.32	[72.28, 74.65]

Table 9: The mean, standard deviation and 95% confidence interval for the mixed approach.

complement such analysis with a comparison between our approach and some state-of-the-art methods, both in terms of plain results and by considering the benefits and disadvantages of the applied techniques.

5.4.1. Effectiveness of the Postulates

The precision and recall metrics indicate the effectiveness of a recommender from two different sides: the quality and the quantity of the recommendations. Over the years, empirical studies have shown that raising a system's recall typically comes at the expense of a decrease in its precision [7, 1]. As can be seen in the performance of our recommender, the precision-recall trade-off is present here as well: those combinations of postulates that result in the lowest precision, are those which achieve the highest recall, while the one with the best precision (Postulate 4) has a very low recall.

Certainly, different application domains will give different precedence to precision and recall. As indicated before, in a medical diagnosis application domain recall is frequently regarded as more important than precision, as it is acceptable to increase the number of false positives if that means to also increase the number of suggested possible causes for a given symptom. This would minimize the possibility of leaving an illness undetected. However, in the movie recommendation domain we regard quality as a more important aspect than quantity. That is, we claim that it is preferable to recommend a small number of movies which the user will most certainly like than to present a large number of unreliable recommendations. Notice that there is no real advantage in recommending to the user hundreds or thousands of movies because users certainly cannot watch them all at the same time. In addition, presenting too many movie recommendations probably means that several of them are wrongly recommended, wasting the users' (movie) time and not fulfilling the system's goal as a recommender. Besides, it can always be the case that after the users have watched the recommended movies they ask for newer recommendations, possibly considering new information (both the rating given to the recommended movies and newer films), assuming that the datasets are updated.

Bearing that setting in mind, we focus the analysis of the performance of the Postulates by prioritizing precision over recall. Moreover, we focus the analysis on positive recommendations (*i.e.*, in favor of watching a movie) rather than on negative ones (*i.e.*, in favor of not watching a movie). The obtained results regarding TP, TN, FP and FN are shown in Figure 6. A comprehensive precision

and recall chart can be seen in Figure 7.

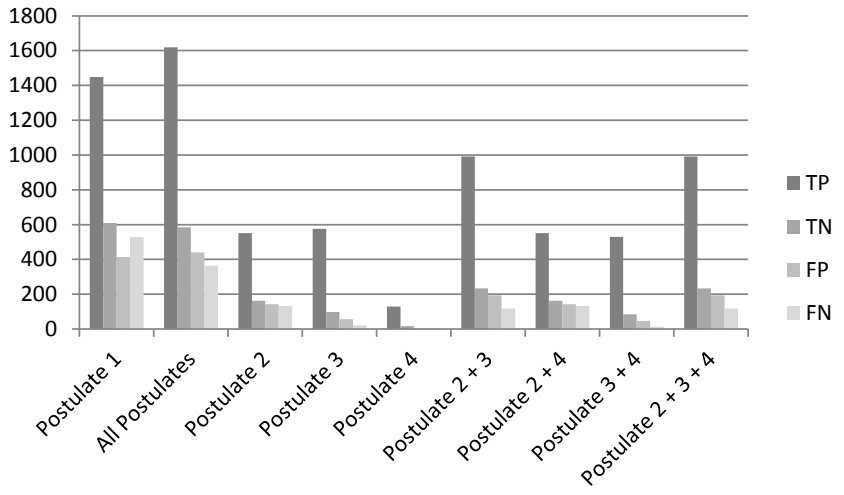


Figure 6: The results obtained by the different combinations of Postulates.

It is important to remark that the performances of all the evaluated postulates is quite good, given that the lowest precision achieved was 0.78. At a first glance, it is evident that Postulate 4 (*i.e.*, recommending a movie to a user based on the user's favorite movie genres and actors *at the same time*) is the most precise approach, and it virtually does not fail when it makes a recommendation, with only five wrong answers among the 141 recommendations given, three of them being false positives (over 132 positive recommendations). Also, notice that the large **UNDECIDED** ratio of Postulate 4 (2849:3000) severely affects its recall, which is explained by the fact that, being the most specific Postulate, it is difficult to find two qualitative reasons (the movie genre and the actors) that simultaneously support the recommendation.

Another important point to notice is that Postulate 3 (which recommends based only on the actors of a movie) has also a really high precision (the difference with Postulate 4 being only 0.07) and greatly improves recall (a difference of 0.22 in favor of Postulate 3). Although we claim that for this application domain precision is more important than recall, this small difference in the quality of the results, along the significant difference in the quantity of recommended results makes Postulate 3 a proper contender to Postulate 4. This is more evident if we consider their $F_{(0.5)}$-score (which praises precision over recall), with a difference of 0.38 in favor of Postulate 3. Additionally, the combination of Postulate 3 and Postulate 4 has also a very good performance, as expected.

As a final remark, we would like to stress out that the difference in the precision between recommending based on the users' favorites actors (Postulate 3) and based on the users' favorites genres (Postulate 2) seems to indicate that the actors performing in a movie are better descriptors of the users' preferences

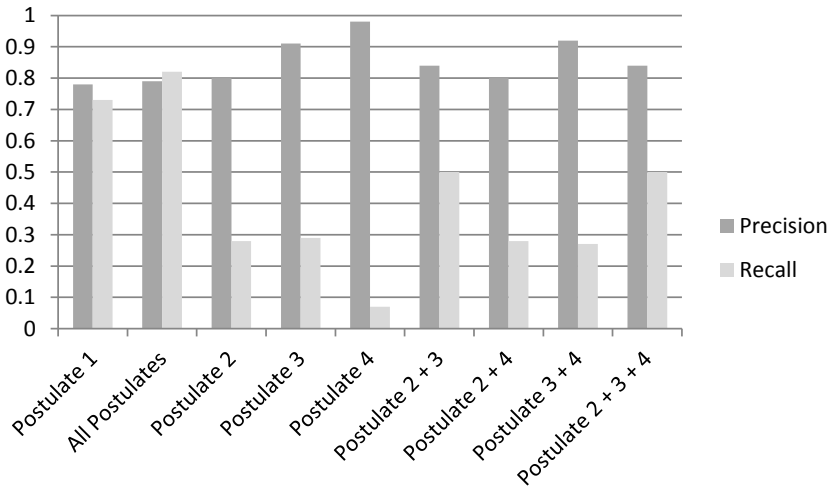


Figure 7: The precision and recall obtained by the different combinations of Postulates.

than a movie's genres.

5.4.2. Comparison of the quantitative approach with the mixed approach

The subsequent part of our analysis intends to find out if there is a significant improvement in the performance of the quantitative approach when we refine its answers by means of qualitative aspects. The first thing interesting to note is that this combination has resulted in an increase in the number of true positives but no other improvements were observed. Thus, it seems that, when it comes to recommendations regarding **not** to watch a movie the difference between them is small, and moreover it is slightly in favor of the quantitative approach. Nevertheless, as pointed out before, the most important recommendations for our movie setting are the positive ones.

We can see that, for the 3000 tested cases, the mixed approach obtains 170 more true positives than the quantitative approach. In addition, the results presented in tables 8 and 9 indicate that the mixed approach results in significant improvements in the number of TP over the quantitative approach. Finally, notice that the improvements are also statistically significant if we consider the entire set of *correct* answers obtained by the approaches, *i.e.*, if we consider the number of TP and TN together, even when that is not the case if we consider the TN rate of the recommenders alone.

The experimental results presented in this section highlight the fact that the incorporation of qualitative aspects in a recommender not only helps to provide transparency but can also result in significant improvements over more basic quantitative approaches.

5.5. Comparison with some state-of-the-art methods used in movie recommendation

In order to complete the analysis of our proposal we provide a comparison between our work and some state-of-the-art approaches to recommendation stemming from quantitative techniques. To evaluate the performance of novel recommendation techniques for the movie domain, most works conduct an offline experiment by using a pre-collected dataset of users' ratings. Two main methodologies for reporting performance are typically adopted: on the one hand, the system's prediction is regarded as a binary classification problem and therefore metrics such as precision, recall and their derivatives are computed. On the other hand, if the system attempts to predict the rating a user would give to a movie, a measure of the accuracy of the system's predicted ratings, such as the root-mean-square error (RMSE), is usually reported.

As has been acknowledged in previous research (*e.g.*, [19]) the second methodology is not suitable for scenarios where only Boolean outcomes (such as positive and negative recommendations) are possible, as it is the case in our argumentation-based recommendation setting. Therefore, we restrict our comparative analysis to works that adopt a precision-oriented evaluation approach. Among these works, we have not found any example in the literature using exactly the same dataset employed in this work, the 10M MovieLens dataset, but rather the experiments were mostly carried out using the 100K and 1M versions. Due to these limitations, a completely fair comparison between our work and other state-of-the-art techniques is not possible. As a consequence, we provide an approximate comparison between our work and three different recent proposals. The first method selected for comparison is based on a novel Bayesian similarity metric [23], the second method is a content-based approach supported by Linked Open Data [14] and the third method is a hybrid approach based on Estimation of Distribution Algorithms [29].

5.5.1. A comparison with a recommender based on a novel Bayesian similarity metric

In [23], a novel Bayesian similarity (BS) approach to measure user similarity is introduced. The proposed similarity measure is based on Dirichlet distribution, taking into consideration both the direction and length of the rating vectors. Such measure helps to overcome several issues coming from computing the cosine similarity or the Pearson correlation coefficient in collaborative filtering approaches. Among the most important benefits of this novel similarity measure is that it diminishes chance correlation (where users are deemed as "similar" only due to a small number of co-rated items). The paper reports an empirical analysis on six real-world datasets, where the experiments indicate that the BS method generates more realistic and distinguishable user similarities than the baseline methods selected for comparison.

In order to compare the BS approach with our approach we focus on the experiments reported by the authors for a movie recommendation scenario based on the 100K version of the MovieLens dataset. It is important to mention

that a direct comparison of the precision values obtained by the BS and our approaches will certainly not be fair, as they present several precision at N (p@n) results, going from 2 to 20 recommendations. Notice that we have not ranked and restricted the number of recommendations considered from the total of 3000 cases, as we only have binary responses, *i.e.*, to recommend or not to recommend. Thus, in a sense, our results can be seen as reporting "precision at rank 3000". While the performance obtained by BS is very high, the precision decreases as the recommendation list grows: while for 2 recommendations the precision obtained is 0.9801, for N = 20 the precision is 0.8338. Thus, considering that we have obtained a precision value of 0.98 for a list of length 3000 (Postulate 4), we can conclude that the performance in terms of precision of our approach is not inferior to that achieved by the BS method. However, as expected, there is a significant difference in terms of recall in favor of the BS approach. Nevertheless, as argued earlier, we consider that in a movie recommendation scenario recall is not a crucial factor, and it may be preferable to trade-off some recall to obtain benefits that could be more significant, such as natural language explanations.

5.5.2. A comparison with a recommender supported by Linked Open Data

A different approach to movie recommendation is taken in [14], where the key idea is to exploit the information encoded in RDF available on the Web of Data. Such information is used to develop a content-based recommender that leverages the data available within Linked Open Data (LOD) datasets, in particular DBpedia, Freebase and LinkedMDB. Based on such information, the authors propose a vector-space-model approach that is used to determine similarities between RDF resources in LOD to generate recommendations. Similarly to our approach, the LOD-based recommender augments the information associated with movies using additional data sources. In this case, the data provided by MovieLens is linked to LOD sources (DBpedia). Another similarity to our proposal is that the LOD-based recommender assumes that if two movies share some information (*e.g.*, part of the cast), then they are related to each other, exploiting such a relation at the moment of defining a user profile. On the other hand, differently from our approach, similarity is determined in a quantitative manner, based on the RDF graph associated with the resources.

The LOD-based recommender is tested using the 1M version of the MovieLens dataset, and as in the BS approach, precision and recall at different ranks are reported. The approach is compared with other non-LOD keyword-based recommender systems, showing that the proposed approach outperforms them. As it is typically the case, precision tends to decrease (and recall to increase) as more recommendations are taken into account. If only the top recommendation is considered (p@1) then the approach achieves a precision of 0.834 with a recall of 0.06. For the maximum computed rank (p@20) the recommender achieves a precision of 0.707, with a perfect recall. Once again, according to the reported results, we can conclude that our approach outperforms the LOD-based approach in terms of precision, and although it has an inferior recall, the achieved results are not substantially inferior than most of those reported in [14].

5.5.3. A comparison with a recommender based on Estimation of Distribution Algorithms

Another approach to movie recommendation is presented in [29], where a novel hybrid recommendation system which takes advantage of collaborative filtering and content-based mechanisms is introduced. A salient feature of this system is the application of the Estimation of Distribution Algorithms (EDAs) to learn users' preferences. The users' preferences are then combined into user interest profiles, which are used to accurately describe users' interest features. Based on the user interest profiles, the content-based mechanism of the model is able to recommend new items to users. In the meantime, the collaborative filtering mechanism suffers less from the sparsity problem because user similarity is determined by the user profiles rather than by the user-item matrix.

The proposed approach is tested using an extension of the 1M version of the MovieLens dataset. Like in our approach, the features associated with each movie are augmented with additional information (directors, writers, actors, genres) obtained from IMDb. The EDA-based approach is compared with other approaches, such as kNN (for k = 30) and Naive Bayes, showing that it outperforms them. The precision at rank 5 achieved by the EDA-based approach is 0.955 while the precision at rank 10 is 0.873. General precision and recall values are also computed with different parameter settings for the EDAs, achieving a precision that fluctuates between 0.736 and 0.772, and a recall that varies from 0.734 to 0.726. Once again we are in the presence of a system whose precision is inferior than that achieved by our approach. On the other hand, the recall obtained by the EDA-based approach is superior to that achieved by our proposal, except when all postulates are used, where our approach achieves a recall of 0.82.

5.5.4. Final comparison

To sum up, Table 10 presents a final comparison of the performance of the analyzed methods. In this table we report the best performance obtained by each of the approaches in terms of precision, also including the number of recommendations for which such result was obtained.

Method	Results @ N	Precision	Recall
All postulates	N = 3000	0.79	0.82
Qualitative postulates (2 + 3 + 4)	N = 3000	0.84	0.50
Postulate 4 alone	N = 3000	0.98	0.07
BS	N = 2	0.98	0.45
LOD	N = 1	0.83	0.06
EDA	N = 5	0.95	0.73

Table 10: Comparison with state-of-the-art methods on movie recommendation.

From the analysis of Table 10 we can conclude that the effectiveness of our

approach is by no means significantly inferior to that of the other approaches. In addition it is important to emphasize the main advantages of our proposal:

- Transparency: although the final users cannot perceive the dialectical process leading to the recommendation (which is clearly not necessary), they have access to clear, natural language explanations of the reasons underlying the given recommendations. This feature helps in building users' trust and improves feedback capabilities.

- Flexibility: the use of rules facilitates the definition of quantitative and qualitative criteria to guide the recommendation process. This is complemented by a simple approach to define argument preference criteria. Moreover, the proposed approach makes it possible to go from a content-based recommender to a collaborative filtering one fairly easily, or even mix them in arbitrary ways, just by switching priorities between rules.

In our opinion, there is no better general approach, and it is the application domain and the users' needs what dictate which is the best method to adopt. For instance, for the movie recommendation scenario the possibility of providing natural language explanations to recommendations seems to be appealing enough even if it comes at the expense of some loss in precision or recall. In other scenarios one could argue that explanations are not that important, while achieving high performance is crucial. In general terms we think that quantitative and qualitative approaches are not at all opposed or excluding each other, but rather complementary. They could greatly benefit from each other, in a similar way as by integrating content-based and collaborative-filtering techniques.

6. Related Work

To the best of the authors' knowledge, this is the first attempt to provide an argument-based approach to movie recommendation with the capability of justifying the suggested items. However, the problem of providing recommendations augmented with justifications has already been studied in previous works. Sinha and Swearingen [43] present an analysis in which they examine the role of transparency in recommender systems, *i.e.*, the importance of a user understanding why a particular recommendation was made. In their work, they present a study that indicates that users prefer transparent recommendations over non-transparent ones. For new items, this result is not surprising. However, the study also shows that users prefer to know why an item was recommended even for those items they already know and like. This suggests that users are not just looking for blind recommendations from a system, but are also looking for a justification of the system's choice.

Tintarev and Masthoff [45] provide a comprehensive review of explanations in recommender systems. In their work they argue that "a good explanation could help inspire user trust and loyalty, increase satisfaction, make it quicker and easier for users to find what they want, and persuade them to try or purchase a recommended item". Furthermore, one can see that explanations must possess

certain characteristics. For instance, one can measure how understandable an explanation is, which can contribute to transparency, user trust, as well as user satisfaction.

Several research works on recommendation systems have focused on the specific domain of movie recommendation. For instance, Basu, Hirsh and Cohen [3] present an approach that uses rating information and user preference to implement a social and content-based movie recommender. In their work they present an inductive learning approach to recommendation, formalizing the recommendation problem as a learning problem. They use *Ripper*, a rule induction system, to learn a function that takes a user and movie and predicts whether the movie will be liked or disliked, combining collaborative and content information. In another work, Melville, P., Mooney, R. J. and Nagarajan, R. [34] propose a framework for implementing a hybrid movie recommender. This approach, called content-boosted collaborative filtering recommender, uses external content information to produce predictions for new users or new items. The content-boosted collaborative filtering recommender has improved prediction performance over some pure content-based recommenders and some pure memory-based collaborative filtering algorithms.

Differently from the above proposals, our work incorporates argument-based inference to the movie recommendation process. In order to support a recommendation for a particular movie, the system performs a qualitative and quantitative analysis on various aspects of a movie (such as the genre, actors, directors, screenwriters, etc.) and on the preferences that a particular user has. Moreover, the recommender not only provides a binary response (recommend or not recommend) but also provides a justification for this answer.

The problem of empowering recommendation with argumentation has been previously addressed in [11]. An argument-based recommendation system for the music domain was introduced in [6]. Further theoretical analyses and simulations aimed at combining quantitative and qualitative aspects to generate argument-based news recommendations were presented in [5]. The work presented here extends these results by describing a novel framework for the specific domain of movie recommendations. Most importantly, it presents an empirical study where classical evaluation metrics are used to determine the effectiveness of the proposal.

7. Conclusions

We have introduced an argumentation-based approach with the capability to improve recommendation technologies. The proposed approach has some strengths that can be proven handy in recommendation scenarios. For instance, argumentation can be used to perform a qualitative analysis on users and items. This allows us to go a step further with respect to the classical approach, giving rise to mixed recommendation systems, *i.e.*, systems that base their recommendations on both qualitative and quantitative aspects of the domain. In particular, this approach can make it easier to take several aspects into consideration before giving the final recommendation to the user. These aspects can

be applied to model some useful features that are sometimes hard to take into account in purely quantitative approaches. For instance, the defeasible nature of users' preferences in complex environments. The quantitative approaches to recommendation are typically based on the use of a number of parameters, which makes the interpretation of the results non intuitive. In our proposal, it is possible to express in a colloquial way the reason behind each recommendation. In addition, we can model restrictions to recommendations based on heterogeneous contextual information (*e.g.*, weather or users' moods).

A useful feature of our formalism is that in order to include a new aspect (postulate) we state it in the form of one or more DeLP rules, and state the preference assigned to these rules. This rule-based modeling of aspects allows to give recommendations based on quantitative criteria (if the rules model them), or qualitative criteria. In the meantime, the use of preferences leads to another advantage: we can go from content-based approaches to collaborative filtering approaches fairly easily, just by switching priorities between rules. Another useful feature is the use of coherent structures of reasons (arguments) to generate explanations for the recommended items. These reasoned explanations are useful in building users' trust and allowing feedback interaction.

We have also provided a comparative analysis between our approach and some state-of-the-art techniques for movie recommendation. It follows from this analysis that the performance of our approach is comparable to that of other existing approaches. Although not superior, it is not substantially inferior either. It is important to emphasize that the main advantages of our approach with respect to other state-of-the-art methods are its transparency (which promotes users' trust) and its flexibility (which facilitates the modeling of different criteria in a natural way).

Finally, it is important to mention that the experiments conducted showed that the argumentation process on which recommendations are based can deal with massive amounts of data, such the 10M MovieLens dataset, as the average response time for the queries was about two seconds. To the best of our knowledge, this is the first time that Defeasible Logic Programming (more precisely, its database-enabled extension) is tested on a very large real-world dataset (such as 10M MovieLens), providing a precedent for its use in massive real-world expert systems.

Future work may be done in different directions. In the first place, we plan to conduct a user study to determine the merits of the explanations presented by the system. Since these explanations are based on rational arguments, they are based on facts and reasons that conform with users' beliefs. As a consequence, we can assume that the given explanations are suitable from a rational perspective. A user study will shed light on interesting issues such as the role of rationality in movie preference. In particular, it could help answer the essential question of whether people are based on rational arguments at the moment of selecting a movie or if irrational or emotional arguments can have a higher influence. On a related subject, we plan to compare the performances of argument-based context-aware recommender systems with the ones proposed in this paper, in order to find out whether or not contextual aspects (*e.g.*, users'

mood, weather, etc.) play an important role in recommendation. Finally, to further elaborate on this last subject we are currently working on an extension of the formalism presented here for modeling context changes through time, to obtain different results for the same query when it is formulated at different time points.

References

[1] Alvarez, S. A., 2002. An exact analytical relation among recall, precision, and classification accuracy in information retrieval. Boston College, Boston, Technical Report BCCS-02-01, 1–22.

[2] Balabanović, M., Shoham, Y., 1997. Fab: content-based, collaborative recommendation. Communications of the ACM 40 (3), 66–72.

[3] Basu, C., Hirsh, H., Cohen, W., et al., 1998. Recommendation as classification: Using social and content-based information in recommendation. In: AAAI/IAAI. pp. 714–720.

[4] Bobadilla, J., Ortega, F., Hernando, A., Gutirrez, A., 2013. Recommender systems survey. Knowledge-Based Systems 46 (0), 109 – 132.

[5] Briguez, C., Capobianco, M., Maguitman, A. G., 2013. A theoretical framework for trust-based news recommender systems and its implementation using defeasible argumentation. International Journal on Artificial Intelligence Tools 22 (4).

[6] Briguez, C. E., Budán, M. C., Deagustini, C. A. D., Maguitman, A. G., Capobianco, M., Simari, G. R., 2012. Towards an argument-based music recommender system. In: Verheij, B., Szeider, S., Woltran, S. (Eds.), COMMA. Vol. 245 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 83–90.

[7] Buckland, M. K., Gey, F. C., 1994. The relationship between recall and precision. JASIS 45 (1), 12–19.

[8] Budán, M. C., Lucero, M. J. G., Chesñevar, C. I., Simari, G. R., 2012. Modelling time and reliability in structured argumentation frameworks. In: KR.

[9] Budzik, J., Hammond, K. J., Birnbaum, L., 2001. Information access in context. Knowledge-based systems 14 (1), 37–53.

[10] Burke, R., 2002. Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction 12 (4), 331–370.

[11] Chesñevar, C., Maguitman, A. G., González, M. P., 2009. Empowering recommendation technologies through argumentation. In: Argumentation in artificial intelligence. Springer, pp. 403–422.

[12] Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., Sartin, M., 1999. Combining content-based and collaborative filters in an online newspaper. In: Proceedings of ACM SIGIR workshop on recommender systems. Vol. 60. Citeseer.

[13] Deagustini, C. A., Dalibón, S. E. F., Gottifredi, S., Falappa, M. A., Chesñevar, C. I., Simari, G. R., 2013. Relational databases as a massive information source for defeasible argumentation. Knowledge-Based Systems 51, 93 – 109.

[14] Di Noia, T., Mirizzi, R., Ostuni, V. C., Romito, D., Zanker, M., 2012. Linked open data to support content-based recommender systems. In: Proceedings of ICSS. ACM, pp. 1–8.

[15] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., 1996. From data mining to knowledge discovery in databases. AI magazine 17 (3), 37.

[16] García, A. J., Simari, G. R., 2004. Defeasible logic programming: An argumentative approach. Theory and practice of logic programming 4 (1+ 2), 95–138.

[17] Goldberg, D., Nichols, D., Oki, B. M., Terry, D., 1992. Using collaborative filtering to weave an information tapestry. Communications of the ACM 35 (12), 61–70.

[18] Grimson, J. B., 1993. Integrating knowledge-based systems and databases. Clinica Chimica Acta 222, 101 – 115.

[19] Grivolla, J., Badia, T., Campo, D., Sonsona, M., Pulido, J.-M., A hybrid recommender combining user, item and interaction data. age 20, 25.

[20] GroupLens, November 2013. Movielens dataset. http://www.grouplens.org/datasets/hetrec-2011.

[21] Grouplens, November 2013. Movielens: Helping you find the right movies. http://movielens.umn.edu.

[22] GroupLens, November 2013. Social computing research at the university of minnesota. http://www.grouplens.org.

[23] Guo, G., Zhang, J., Yorke-Smith, N., 2013. A novel bayesian similarity measure for recommender systems. In: Proceedings of the Twenty-Third international joint conference on Artificial Intelligence. AAAI Press, pp. 2619–2625.

[24] IMDb.com, I., November 2013. The internet movie database. http://www.imdb.com.

[25] Kowalski, R., 1974. Predicate logic as programming language. In: IFIP congress.

[26] Kurgan, L. A., Musilek, P., 2 2006. A survey of knowledge discovery and data mining process models. The Knowledge Engineering Review 21, 1–24.

[27] Laenens, E., Vermeir, D., 1990. Advanced knowledge-base environments for large database systems. Knowledge-Based Systems 3 (4), 215–220.

[28] Lee, J. S., Lee, J. C., 2007. Context awareness by case-based reasoning in a music recommendation system. In: Ichikawa, H., Cho, W.-D., Satoh, I., Youn, H. Y. (Eds.), UCS. Vol. 4836 of Lecture Notes in Computer Science. Springer, pp. 45–58.

[29] Liang, T., Liang, Y., Fan, J., Zhao, J., 2014. A hybrid recommendation model based on estimation of distribution algorithms. Journal of Computational Information Systems 10 (2), 781–788.

[30] LiKamWa, R., Liu, Y., Lane, N. D., Zhong, L., 2013. Moodscope: building a mood sensor from smartphone usage patterns. In: Chu, H.-H., Huang, P., Choudhury, R. R., Zhao, F. (Eds.), MobiSys. ACM, pp. 465–466.

[31] Linton, F., Joy, D., Schaefer, H.-P., 1999. Building user and expert models by long-term observation of application usage. Courses and Lectures-International Centre for Mechanical Sciences, 129–138.

[32] Maes, P., et al., 1994. Agents that reduce work and information overload. Communications of the ACM 37 (7), 30–40.

[33] Martinez, M. V., García, A. J., Simari, G. R., 2012. On the use of presumptions in structured defeasible reasoning. In: Fourth International Conference on Computational Models of Argument (COMMA 2012). pp. 185–196.

[34] Melville, P., Mooney, R. J., Nagarajan, R., 2002. Content-boosted collaborative filtering for improved recommendations. In: AAAI/IAAI. pp. 187–192.

[35] Michie, D., 1990. Machine learning: principles and techniques : Richard forsyth. Knowledge Based Systems 3 (1), 59.

[36] Pazzani, M. J., Billsus, D., 1999. Evaluating adaptive web site agents. In: Proc. Workshop on Recommender Systems Algorithms and Evaluation, 22nd International Conference on Research and Development in Information Retrieval.

[37] Pazzani, M. J., Billsus, D., 2007. Content-based recommendation systems. In: The adaptive web. Springer, pp. 325–341.

[38] Rahwan, I., Simari, G. R., 2009. Argumentation in Artificial Intelligence. Springer.

[39] Ramakrishnan, R., Ullman, J. D., 1993. A survey of research on deductive database systems. Journal of Logic Programming 23, 125–149.

[40] Sandvig, J. J., Mobasher, B., Burke, R. D., 2008. A survey of collaborative recommendation and the robustness of model-based algorithms. IEEE Data Eng. Bull. 31 (2), 3–13.

[41] Sarwar, B., Karypis, G., Konstan, J., Riedl, J., 2001. Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web. ACM, pp. 285–295.

[42] Simari, G. R., Loui, R. P., 1992. A mathematical treatment of defeasible reasoning and its implementation. Artificial Intelligence 53 (2-3), 125–157.

[43] Sinha, R., Swearingen, K., 2002. The role of transparency in recommender systems. In: CHI'02 extended abstracts on Human factors in computing systems. ACM, pp. 830–831.

[44] Stolzenburg, F., García, A. J., Chesnevar, C. I., Simari, G. R., 2003. Computing generalized specificity. Journal of Applied Non-Classical Logics 13 (1), 87–113.

[45] Tintarev, N., Masthoff, J., 2007. A survey of explanations in recommender systems. In: Data Engineering Workshop, 2007 IEEE 23rd International Conference on. IEEE, pp. 801–810.

[46] Williams, M., Chen, G., Ferbrache, D., Massey, P., Salvini, S., Taylor, H., Wong, K., 1988. Prolog and deductive databases. Knowledge-Based Systems 1 (3), 188 – 192.