Argument-Based Critics and Recommenders: A Qualitative Perspective on User Support Systems

Carlos Iván Chesñevar [a] Ana Gabriela Maguitman [b]
Guillermo Ricardo Simari [c]

[a] *Department of Computer Science*
Universitat de Lleida – C/Jaume II, 69 – 25001 Lleida, Spain
Email: cic@eps.udl.es

[b] *School of Informatics*
Indiana University – Bloomington, IN 47408-3912 , USA
Email: anmaguit@cs.indiana.edu

[c] *Dept. of Computer Science and Engineering*
Universidad Nacional del Sur – 8000 B.Blanca, Argentina
Email: grs@cs.uns.edu.ar

Abstract

In recent years we have witnessed the wide-spread evolution of support tools that operate in association with the user to accomplish a range of computer-mediated tasks. Two examples of these tools are critics and recommenders. Critics are cooperative tools that observe the user interacting with a computer system and present reasoned opinions about a product under development. Recommender systems are tools that assist users by facilitating access to relevant items. At the same time, defeasible argumentation has evolved as a successful approach in AI to model commonsense qualitative reasoning, with applications in many areas, such as agent theory, knowledge engineering and legal reasoning. This paper presents a novel approach towards the integration of user support systems, such as critics and recommender systems, with a defeasible argumentation framework. The final goal is to enhance practical reasoning capabilities of current user support tools by incorporating argument-based qualitative inference.

Key words: User Support Systems, Decision support systems, Recommender systems, Critics, Defeasible Argumentation, Practical reasoning

1 Introduction and Motivations

Critiquing and recommendation systems have evolved in the last years as specialized tools to assist users in a plethora of computer-mediated tasks by providing guidelines or hints [93,56]. Critiquing systems (also known as *critics*) offer context-sensitive support to users. The goal of critiquing systems is to discover and point out errors or suboptimal results that might otherwise remain unnoticed, and to help users make the necessary repairs. Recommender systems (or just *recommenders*), on the other hand, are aimed at helping users with the problem of information overload by facilitating access to relevant items [67]. In a more general setting, both critics and recommenders can be conceptualized as *user support systems.*

Most critics and recommenders are based on machine learning and information retrieval algorithms. The resulting systems typically provide suggestions based on *quantitative* evidence (i.e., measures of similarity between objects or users). The inference process which led to these suggestions is commonly opaque to the user (i.e., 'black-box' metaphor). Although the effectiveness of existing critics and recommenders is remarkable, they still have serious limitations as they are unable to perform qualitative inference on the suggestions they offer. A number of well-know formalisms, such as description logics [38], could be naturally integrated to existing recommendation technologies to provide inference capabilities. However, the majority of these formalisms are incapable of dealing with the defeasible nature of user preferences. In this context, *defeasible argumentation* frameworks constitute an interesting alternative, as they have matured in the last decade to become a useful setting to formalize commonsense qualitative reasoning in a computationally attractive way. A thorough consideration of these frameworks can be found in [22,87].

Recent research has shown that argumentation can be integrated in a growing number of real-world applications such as multiagent systems [77,2,76], legal reasoning [86], knowledge engineering [17], analysis of news reports [51] clustering [48], argumentation support systems [107], mediation systems and computer-supported collaborative argumentation [68,89,49,66], among others. In the last few years particular attention has been given to different extensions of *logic programming* as a suitable framework for formalizing knowledge representation and argumentative inference.

Defeasible logic programming (DeLP) [44][1] is a particular formalization of defeasible argumentation based on logic programming which provides the possibility of representing information in the form of defeasible and strict rules in a declarative manner. An important characteristic of the DeLP approach is

[1] An implementation of DeLP is available at `http://lidia.cs.uns.edu.ar/DeLP`.

that, by performing defeasible reasoning dialectically, it can handle information that is in principle contradictory. The process of deciding if a conclusion is supported, or *warranted*, begins by analyzing if there exists an undefeated argument (a *warrant*) supporting that conclusion, i.e, an argument for which every possible attacking argument has been defeated. The notion of *attack* and *defeat* is formally introduced below but can be intuitively described as the consideration of arguments that are in conflict (attack) with the supporting argument. The attack becomes a defeat when the attacking argument is better, in some specific sense, than the supporting argument.

Thus, use of argumentation will allow the system to present reasoned suggestions, which the user will be able to further investigate and accept only if a convincing case can be made by the recommendation tool. This approach has proven to be particularly attractive in the context of real-world applications, such as clustering [48], intelligent web search [21], natural language processing [20], and multiagent systems [25]. An in depth analysis of DeLP and its comparison with other formalisms in argumentation and logic programming can be found in [44].

This paper presents a first approach towards the integration of typical variants of user support systems –such as critics and recommenders– with a defeasible argumentation framework. In order to enhance user support by means of argumentation, user preference criteria are modelled by means of facts, strict rules and defeasible rules encoded as part of a DeLP program. These preference criteria are combined with additional background information and used by the argumentation framework to prioritize potential suggestions, thus enhancing the final results provided to the active user.

The rest of the article is structured as follows. First, in Section 2 we present a generic characterization of user support systems, with focus on critics and recommenders. In Section 3 we provide some background on the role of argumentation in AI and outline the fundamentals of defeasible logic programming. Next, in Section 4 we discuss our proposal for characterizing argument-based recommender systems. Sections 5 and 6 present two case studies, which illustrate how the proposed approach works. Section 7 describes implementation issues and ongoing research. Section 8 discusses related work. Finally, in Section 9 we present the main conclusions that have been obtained.

2 User Support Systems

User support system operate in association with the user to effectively accomplish a range of tasks. Some of these systems serve the purpose of expanding the user's natural capabilities, for example by acting as intelligence or memory augmentation mechanisms [37]. Some of these systems reduce the user's work by carrying out the routinizable tasks on the user's behalf. Others offer tips on how to refine or complete human generated products (such as electronic documents) by highlighting potential inaccuracies and proposing alternative solutions. Some aides "think ahead" to anticipate the next steps in a user's task providing the capability for the user to confirm the prediction and ask the system to complete the steps automatically.

As we will see later, argument-based reasoning can be integrated into such user support systems in order to provide a qualitative perspective in decision making. This is achieved by integrating inference abilities to offer reasoned suggestions and critics modelled in terms of arguments in favor and against a particular decision. This approach complements existing qualitative techniques by enriching the user's mental model of such computer systems in a natural way: suggestions are statements which are backed by arguments supporting them. Clearly, conflicting suggestions may arise, and it will be necessary to determine which suggestions can be considered as valid according to some rationally justified procedure. The role of argumentation is to provide a sound formal framework as a basis for such analysis.

Next we will briefly review two major kinds of user support systems –critics and recommenders– which cover an important part of the spectrum of existing user support system technologies. Both critics and recommenders act in cooperation with users, complementing their abilities and augmenting their performance by offering proactive or on demand context-sensitive support.

2.1 Critics

Critics or *critiquing systems* [98] are cooperative tools that observe the user interacting with a computer system and present reasoned opinions about a product under development. The goal of the critiquing systems is to discover and point out errors or suboptimal results that might otherwise remain unnoticed, and to help users to make the necessary repairs. Critics need domain-based criteria to evaluate the quality of a solution and usually generate their advice by using a specialized domain knowledge base. Critics have been implemented in many applications, such as tutoring [97], diagnosis and decision-making [73], expertise-based design [40], and knowledge acquisition systems [104]. Most popular critiquing systems have been developed to assist *word processing.* These include spelling-, grammar-, and style-checkers [59,32,88].

Although critics for word-processing software have helped to significantly reduce the overall amount of burden for checking documents, the problem of judging the appropriateness of language usage in different contexts remains to a large extent still unsolved. Providing advice on language usage is hard for several reasons. On the one hand, language usage does not follow a set of predefined and rigorous rules, being highly context-dependant (thus, for example, some colloquial words in British English could lead to misunderstandings when used for an American audience). On the other hand, natural languages change over time, and such changes turn out to be very difficult to trace by means of automated tools.

One effective solution to this problem is to provide the user with information about frequencies of natural language expressions in different contexts. Such systems, called *concordance programs* [41,91], have become particularly powerful with the evolution of the web, as there is a huge collection of text-based information available online. However, there is still a major problem with this approach, as concordance programs require the user to *explicitly* provide a search-string and for that reason they have a limited functionality. Critiquing systems with inference abilities for judging the appropriateness of language usage could automate this process, reducing the user's effort in identifying inaccuracies and generating solutions. As we will see in Section 6, the combination of automated argument-based inference along with Web search capabilities will provide a powerful tool for reasoning about appropriate language usage.

2.2 Recommenders

Recommender systems are programs that create a model of the user's preferences or the user's task with the purpose of facilitating access to items (e.g., news, web pages, books, etc.) that the user may find useful [93,56]. While in many situations the user explicitly posts a request for recommendations in the form of a query, many recommender systems attempt to anticipate the user's needs and are capable of proactively providing assistance [94,95,14]. In order to come up with recommendations for user queries, conventional recommender systems rely on *similarity measures* between users or contents, computed on the basis of methods coming either from the information retrieval or the machine learning communities. Recommender systems adopt mainly two different views to help predict information needs. The first approach is known as *user modelling* and relies on the use of a profile or model of the users, which is created by observing users' behavior (e.g., [65,34]). The second approach is based on *task modelling*, and recommendations are based on the context in which the user is immersed (e.g., [15,61]). 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 are driven by the premise that user's preferences tend to persist through time. These recommenders frequently use machine-learning techniques to induce a profile of the active user. Typically, a model of the active user is 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 are based 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 (e.g., [7,6,12,33]). In particular, given the huge amount of information existing on the Web, it is not surprising that the great majority of the recommender systems have been built around content and resources available online (e.g., [3,74,62,78,35]).

One of the main issues faced by recommender systems is the "sparsity problem". This problem is due to the reluctance of users to rate items. Therefore, much research has focused on the development of strategies for filling in incomplete users' models. A common technique is to monitor the users' behavior. [2] While this approach minimizes the user's effort and generates large amounts of data, it has the disadvantage of resulting in noisy descriptions of the user's interests. Another issue faced by recommendation systems is that of trustworthiness. This is sometimes due to the absence of any rationale supporting the presented suggestions. Therefore, the user is unable to evaluate the reasons that led the system to present certain recommendations. In certain domains (e.g., e-commerce), this lack of justification can be associated with ulterior motives on the recommendation provider's side, leading to lack of confidence or reliability. Typical approaches to recommendation (especially those based on IR and machine learning techniques) are usually limited in this sense. As we will later see in Section 5, the use of argumentation will allow to enhance recommender systems with inference abilities to present reasoned suggestions, which the user will be able to further investigate and accept only if a convincing case can be made by the recommendation tool.

[2] See for example the Amazon recommendation system for books (http://amazon.com).

3 Modelling Argumentation in DeLP

3.1 Argumentation in AI: Background

Artificial Intelligence (AI) has long dealt with the enormous challenge of modelling commonsense reasoning, which almost always occurs in the face of incomplete and potentially inconsistent information [70,90,69]. A logical model of commonsense reasoning demands the formalization of principles and criteria that characterize valid patterns of inference. In this respect, classical logic has proven to be inadequate, since it behaves *monotonically* and cannot deal with inconsistencies at object level [90].

When a rule supporting a conclusion may be defeated by new information, it is said that such reasoning is *defeasible* [80,81,75,82,99]. When we chain defeasible reasons or rules to reach a conclusion, we have *arguments* instead of proofs. Arguments may compete, rebutting each other, so a *process* of argumentation is a natural result of the search for arguments. Adjudication of competing arguments must be performed, comparing arguments in order to determine what beliefs are ultimately accepted as *warranted* or *justified*. Preference among conflicting arguments is defined in terms of a *preference criterion* which establishes a relation " $\preceq$ " among possible arguments; thus, for two arguments A and B in conflict, it may be the case that A is strictly preferred over B ($A \succ B$), that A and B are equally preferable ($A \succeq B$ and $A \preceq B$) or that A and B are not comparable with each other. In the above setting, since we arrive at conclusions by building defeasible arguments, and since *logical argumentation* is usually called *argumentation*, we sometimes call this kind of reasoning *defeasible argumentation*.

For the sake of example, let us consider a well-known problem of nonmonotonic reasoning in AI about the flying abilities of birds, recast in argumentative terms. Consider the following sentences:

(1) Birds usually fly.
(2) Penguins usually do not fly.
(3) Penguins are birds.

The first two sentences correspond to *defeasible rules* (rules which are subject to possible exceptions). The third sentence is a *strict rule*, where no exceptions are possible. Now, given the fact that *Tweety is a penguin* two different arguments can be constructed:

(1) Argument A (based on rules 1 & 3): Tweety is a penguin. Penguins are birds. Birds usually fly. So Tweety flies.
(2) Argument B (based on rule 2): Tweety is a penguin. Penguins usually do

not fly. So Tweety does not fly.

In this particular situation, two arguments arise that cannot be accepted simultaneously (as they reach contradictory conclusions). Note that argument B seems rationally preferable over argument A, as it is based on more *specific* information. As a matter of fact, specificity is commonly adopted as a syntax-based criterion among conflicting arguments, preferring those arguments which are *more informed* or *more direct* [84,83]. In this particular case, if we adopt specificity as a preference criterion, argument B is justified, whereas A is not (as it is defeated by B). The above situation can easily become much more complex, as an argument may be defeated by a second argument, which in turn can be defeated by a third argument, *reinstating* the first one.

Several defeasible argumentation frameworks have been developed on the basis of extensions to logic programming (see [22,87,54]). *Defeasible logic programming* (DeLP) [44] is one of such formalisms, combining results from defeasible argumentation theory [99] and logic programming. DeLP is a suitable framework for building real-world applications that deal with incomplete and contradictory information in dynamic domains. In what follows we will present a brief overview of the DeLP framework. A more in-depth treatment can be found elsewhere [44,28].

3.2 Defeasible Logic Programming

A defeasible logic program is a set (Π, Δ) of rules, where Π and Δ stand for sets of *strict* and *defeasible* knowledge, resp. The set Π involves *strict rules* of the form $P \leftarrow Q_1, \ldots, Q_k$ and *facts* (strict rules with empty body), and it is assumed to be *non-contradictory*. The set Δ involves *defeasible rules* of the form $P \mathrel{-\!\!\prec} Q_1, \ldots, Q_k$. The underlying logical language is that of extended logic programming [45,64], enriched with a special symbol "$\mathrel{-\!\!\prec}$" to denote defeasible rules. Both default and classical negation are allowed (denoted not and $\sim$, resp.).[3] Literals preceded by not are called *extended literals* [45]. DeLP rules are to be thought of as *inference rules* rather than implications in the object language.

Example 1 (Adapted from [30]) *Consider an intelligent agent controlling an engine with an oil pump, a fuel pump and an engine, as well as three switches* $sw1$, $sw2$ *and* $sw3$. *These switches regulate different features of the engine, such as the pumping system, speed, etc. This agent may have the following defeasible knowledge base for diagnosing possible problems with the engine:*

[3] The definitions that follow summarize DeLP with default negation (see discussion in [44, pages 30-33]).

$$\begin{array}{lr}
(1) & \sim fuel\_ok \;\leftarrow\; pump\_clog \\
(2) & sw1 \\
(3) & sw2 \\
(4) & sw3 \\
(5) & heat \\
(6) & pump\_fuel \mathrel{-\!\!\prec} sw1 \\
(7) & fuel\_ok \mathrel{-\!\!\prec} pump\_fuel \\
(8) & pump\_oil \mathrel{-\!\!\prec} sw2 \\
(9) & oil\_ok \mathrel{-\!\!\prec} pump\_oil \\
(10) & engine\_ok \mathrel{-\!\!\prec} fuel\_ok, oil\_ok \\
(11) & \sim engine\_ok \mathrel{-\!\!\prec} heat \\
(12) & \sim oil\_ok \mathrel{-\!\!\prec} heat \\
(13) & pump\_clog \mathrel{-\!\!\prec} pump\_fuel, low\_speed \\
(14) & low\_speed \mathrel{-\!\!\prec} sw2 \\
(15) & \sim low\_speed \mathrel{-\!\!\prec} sw2, sw3 \\
(16) & fuel\_ok \mathrel{-\!\!\prec} sw3
\end{array}$$

Fig. 1. DeLP program $\mathcal{P}_{eng}$ (example 1)

– *If the pump is clogged, then the engine gets no fuel.*
– *When* $sw1$ *is on, normally fuel is pumped properly.*
– *When fuel is pumped properly, fuel seems to work ok.*
– *When* $sw2$ *is on, usually oil is pumped.*
– *When oil is pumped, usually it works ok.*
– *When there is oil and fuel, usually the engine works ok.*
– *When there is heat, then the engine is usually not ok.*
– *When there is heat, normally there are oil problems.*
– *When fuel is pumped and speed is low, then there are reasons to believe that the pump is clogged.*
– *When* $sw2$ *is on, usually speed is low.*
– *When* $sw2$ *and* $sw3$ *are on, usually speed is not low.*
– *When* $sw3$ *is on, usually fuel is ok.*
Suppose also that the agent knows some particular facts: $sw1$, $sw2$ *and* $sw3$ *are on, and there is heat. The knowledge of such an agent can be modelled by the DeLP program* $\mathcal{P}_{eng} = (\Pi_{eng}, \Delta_{eng})$ *shown in Fig. 1, where the set* Π_{eng} *of strict knowledge corresponds to clauses* $(1)-(4)$, *and the set* Δ_{eng} *of defeasible knowledge corresponds to clauses* $(5)-(16)$.

Deriving literals in DeLP results in the construction of *arguments*. Formally:

Definition 2 (Argument) *Given a DeLP program* $\mathcal{P}$, *and let* Q *be a literal in* $\mathcal{P}$. *An* argument $\mathcal{A}$ *for the query* Q, *denoted* $\langle \mathcal{A}, Q \rangle$, *is a subset of ground instances of defeasible rules in* $\mathcal{P}$ *such that:*

(1) there exists a defeasible derivation *for* Q *from* $\Pi \cup \mathcal{A}$;
(2) $\Pi \cup \mathcal{A}$ *is non-contradictory (i.e,* $\Pi \cup \mathcal{A}$ *does not entail two complementary literals* P *and* $\sim P$, *nor does* $\mathcal{A}$ *contain literals* S *and* not S)[4] *and*

[4] We use the term "contradictory" instead of "inconsistent" to avoid confusion, as

(3) $\mathcal{A}$ is the minimal set (with respect to set inclusion) satisfying (1) and (2).

An argument $\langle \mathcal{A}_1, Q_1 \rangle$ is a sub-argument *of another argument $\langle \mathcal{A}_2, Q_2 \rangle$ if $\mathcal{A}_1 \subseteq \mathcal{A}_2$. Given a DeLP program $\mathcal{P}$, $Args(\mathcal{P})$ denotes the set of all possible arguments that can be derived from $\mathcal{P}$.*

The notion of defeasible derivation corresponds to the usual query-driven SLD derivation used in logic programming, performed by backward chaining on both strict and defeasible rules, but only "collecting" defeasible rules as part of the argument. In this context a negated literal $\sim P$ is treated just as a new predicate name no_P. Minimality imposes the 'Occam's razor principle' [99] on arguments. Any superset $\mathcal{A}'$ of $\mathcal{A}$ can be proven to be 'weaker' than $\mathcal{A}$ itself, as the former relies on more defeasible information. The non-contradiction requirement forbids the use of (ground instances of) defeasible rules in an argument $\mathcal{A}$ whenever $\Pi \cup \mathcal{A}$ entails complementary literals. It should be noted that non-contradiction captures the two usual approaches to negation in logic programming (viz. default negation and classic negation), both of which are present in DeLP and are related to the notion of counterargument, as we will see in the next Definition.

Example 3 *Consider the program $\mathcal{P}_{eng}$ in Example 1. Arguments $\langle \mathcal{B}, fuel\_ok \rangle$ and $\langle \mathcal{C}, oil\_ok \rangle$ can be derived from $\mathcal{P}_{eng}$, with*

$$\mathcal{B} = \{pump\_fuel \prec sw1\,;\, fuel\_ok \prec pump\_fuel\}$$
$$\mathcal{C} = \{pump\_oil \prec sw2\,; oil\_ok \prec pump\_oil\}.^{5}$$

Similarly, an argument $\langle \mathcal{A}_1, engine\_ok \rangle$ can be derived from $\mathcal{P}_{eng}$, where

$$\mathcal{A}_1 = \{engine\_ok \prec fuel\_ok, oil\_ok\} \cup \mathcal{B} \cup \mathcal{C}$$

Note that in this last case $\langle \mathcal{C}, oil\_ok \rangle$ and $\langle \mathcal{B}, fuel\_ok \rangle$ are subarguments *of $\langle \mathcal{A}_1, engine\_ok \rangle$.*

Definition 4 (Counterargument – Defeat) *An argument $\langle \mathcal{A}_1, Q_1 \rangle$ is a* counterargument *for an argument $\langle \mathcal{A}_2, Q_2 \rangle$ (or equivalently $\langle \mathcal{A}_1, Q_1 \rangle$ counterargues $\langle \mathcal{A}_2, Q_2 \rangle$) if and only if*

(1) There is an subargument $\langle \mathcal{A}, Q \rangle$ of $\langle \mathcal{A}_2, Q_2 \rangle$ such that the set $\Pi \cup \{Q_1, Q\}$ is contradictory, or
(2) An extended literal not Q_1 *is present in some rule in $\mathcal{A}_2$.*[6]

the latter is commonly used in the context of classical logic.

[5] For the sake of clarity, we use semicolons to separate elements in an argument $\mathcal{A} = \{e_1\ ;\ e_2\ ;\ \ldots ;\ e_k\ \}$.

[6] The first notion of attack is borrowed from the Simari-Loui framework [99]; the second one is related to Dung's argumentative approach to logic programming [36] as well as to other formalizations, such as [85,58]. For an in-depth discussion see [44].

A preference criterion $\preceq \subseteq Args(\mathcal{P}) \times Args(\mathcal{P})$ will be used to decide among conflicting arguments. An argument $\langle \mathcal{A}_1, Q_1 \rangle$ is a defeater *for an argument $\langle \mathcal{A}_2, Q_2 \rangle$ if $\langle \mathcal{A}_1, Q_1 \rangle$ counterargues $\langle \mathcal{A}_2, Q_2 \rangle$, and $\langle \mathcal{A}_1, Q_1 \rangle$ is preferred over $\langle \mathcal{A}_2, Q_2 \rangle$ with respect to $\preceq$. For cases (1) and (2) above, we distinguish between* proper *and* blocking defeaters *as follows:*

- *In case (1), the argument $\langle \mathcal{A}_1, Q_1 \rangle$ will be called a* proper defeater *for $\langle \mathcal{A}_2, Q_2 \rangle$ if and only if $\langle \mathcal{A}_1, Q_1 \rangle$ is strictly preferred over $\langle \mathcal{A}, Q \rangle$ with respect to $\preceq$.*
- *In case (1), if $\langle \mathcal{A}_1, Q_1 \rangle$ and $\langle \mathcal{A}, Q \rangle$ are unrelated to each other with respect to $\preceq$, or in case (2), $\langle \mathcal{A}_1, Q_1 \rangle$ will be called a* blocking defeater *for $\langle \mathcal{A}_2, Q_2 \rangle$.*

It must be noted that in DeLP there is no explicit distinction between undercutting and rebutting defeat [80,81], as both of them are subsumed by the notion of counterargument. The notions of proper and blocking defeat in Def. 4 distinguish between defeaters which are "strictly better" and defeaters which are "as good as" the argument under attack, respectively.

Example 5 *Consider the argument $\langle \mathcal{A}_1, engine\_ok \rangle$ given in Example 3 wrt the program $\mathcal{P}_{eng}$. A counteragument for $\langle \mathcal{A}_1, engine\_ok \rangle$ can be found, namely $\langle \mathcal{A}_2, {\sim}fuel\_ok \rangle$, with*

$$\begin{aligned} \mathcal{A}_2 \quad = \quad \{ \ & pump\_fuel \prec sw1,\ low\_speed \prec sw2, \\ & pump\_clog \prec pump\_fuel, low\_speed\} \ \} \end{aligned}$$

Argument $\langle \mathcal{A}_2, {\sim}fuel\_ok \rangle$ is a counterargument for $\langle \mathcal{A}_1, engine\_ok \rangle$ as there exists a subargument $\langle \mathcal{B}, fuel\_ok \rangle$ in $\langle \mathcal{A}_1, engine\_ok \rangle$ (see Example 3) such that $\Pi_{eng} \cup \{fuel\_ok, {\sim}fuel\_ok\}$ is contradictory.

Specificity [99] is used in DeLP as a syntactic preference criterion among conflicting arguments, favoring those arguments that are *more informed* or *more direct* [99,102]. However, other alternative preference criteria could also be used [44].

Example 6 *Consider the arguments $\langle \mathcal{A}_1, engine\_ok \rangle$ and $\langle \mathcal{A}_2, {\sim}fuel\_ok \rangle$ in Example 5. Then $\langle \mathcal{A}_2, {\sim}fuel\_ok \rangle$ is a proper defeater for $\langle \mathcal{A}_1, engine\_ok \rangle$, as $\langle \mathcal{A}_2, {\sim}fuel\_ok \rangle$ counterargues $\langle \mathcal{A}_1, engine\_ok \rangle$ with disagreement subargument $\langle \mathcal{B}, fuel\_ok \rangle$, and $\langle \mathcal{A}_2, {\sim}fuel\_ok \rangle$ is strictly more specific than $\langle \mathcal{B}, fuel\_ok \rangle$.*

Given an argument $\langle \mathcal{A}, Q \rangle$, the definitions of counterargument and defeat allows to detect whether other possible arguments $\langle \mathcal{B}_1, Q_1 \rangle, \ldots, \langle \mathcal{B}_k, Q_k \rangle$ are defeaters for $\langle \mathcal{A}, Q \rangle$. Should the argument $\langle \mathcal{A}, Q \rangle$ be defeated, then it would be no longer supporting its conclusion Q. However, since defeaters are arguments, they may on their turn be defeated. That prompts for a complete recursive dialectical analysis to determine which arguments are ultimately defeated. To characterize this process we will introduce some auxiliary notions.

An *argumentation line* starting in $\langle \mathcal{A}_0, Q_0 \rangle$ (denoted $\lambda^{\langle \mathcal{A}_0, Q_0 \rangle}$) is a sequence $[\langle \mathcal{A}_0, Q_0 \rangle, \langle \mathcal{A}_1, Q_1 \rangle, \langle \mathcal{A}_2, Q_2 \rangle, \ldots, \langle \mathcal{A}_n, Q_n \rangle, \ldots]$ that can be thought of as an exchange of arguments between two parties, a *proponent* (evenly-indexed arguments) and an *opponent* (oddly-indexed arguments). Each $\langle \mathcal{A}_i, Q_i \rangle$ is a defeater for the previous argument $\langle \mathcal{A}_{i-1}, Q_{i-1} \rangle$ in the sequence, $i > 0$. In order to avoid *fallacious* reasoning, dialectical constraints are imposed on such an argument exchange to be considered rationally acceptable in light of a given program $\mathcal{P}$, namely:

1) **Non-contradiction:** given an argumentation line λ, the set of arguments of the proponent (resp. opponent) should be *non-contradictory* wrt $\mathcal{P}$.
2) **No circular argumentation:** given an argumentation line λ, no argument $\langle \mathcal{A}_j, Q_j \rangle$ is allowed to appear as a sub-argument of another argument $\langle \mathcal{A}_i, Q_i \rangle$ $i < j$.
3) **Progressive argumentation:** every blocking defeater $\langle \mathcal{A}_i, Q_i \rangle$ in λ is defeated by a proper defeater $\langle \mathcal{A}_{i+1}, Q_{i+1} \rangle$ in λ.

The first condition disallows the use of contradictory information on either side (proponent or opponent). The second condition eliminates the "circular reasoning" fallacy. The last condition enforces the use of a stronger argument to defeat an argument which acts as a blocking defeater. An argumentation line satisfying the above restrictions is called *acceptable*, and can be proven to be finite. It must be noted that other argumentative frameworks (e.g. [11,54,109,24]) introduce similar constraints on argumentation lines to avoid infinite "chains" of defeaters, reciprocal defeaters, etc. A more detailed analysis on such situations can be found in [87].

Example 7 *Consider the argument $\langle \mathcal{A}_1, engine\_ok \rangle$ and the associated defeater $\langle \mathcal{A}_2, {\sim}fuel\_ok \rangle$ in Example 6. Note that $\langle \mathcal{A}_2, {\sim}fuel\_ok \rangle$ has the associated subargument $\langle \mathcal{A}_2{}', low\_speed \rangle$, with $\mathcal{A}_2{}' = \{low\_speed \prec sw2\}$. From the program $\mathcal{P}_{eng}$ (Fig. 1) a blocking defeater for $\langle \mathcal{A}_2, {\sim}fuel\_ok \rangle$ can be derived, namely $\langle \mathcal{A}_3, {\sim}low\_speed \rangle$. Note that this third defeater can be thought of as an answer of the proponent to the opponent,* reinstating *the first argument $\langle \mathcal{A}_1, engine\_ok \rangle$, as it defeats the opponent's defeater $\langle \mathcal{A}_2, {\sim}fuel\_ok \rangle$. The above situation can be expressed in the following argumentation line:*

$$[\langle \mathcal{A}_1, engine\_ok \rangle, \langle \mathcal{A}_2, {\sim}fuel\_ok \rangle, \langle \mathcal{A}_3, {\sim}low\_speed \rangle].$$

Note that the proponent's last defeater in the above sequence could be on its turn defeated by a blocking defeater $\langle \mathcal{A}_2{}', low\_speed \rangle$, resulting in

$$[\langle \mathcal{A}_1, engine\_ok \rangle, \langle \mathcal{A}_2, {\sim}fuel\_ok \rangle, \langle \mathcal{A}_3, {\sim}low\_speed \rangle, \langle \mathcal{A}_2{}', low\_speed \rangle \ldots]$$

However, such line is not acceptable*, as it violates the condition of non-circular argumentation.*

Given a program $\mathcal{P}$ and an initial argument $\langle \mathcal{A}_0, Q_0 \rangle$, the set of all acceptable argumentation lines starting in $\langle \mathcal{A}_0, Q_0 \rangle$ accounts for a whole dialectical analysis for $\langle \mathcal{A}_0, Q_0 \rangle$ (i.e., all possible dialogues rooted in $\langle \mathcal{A}_0, Q_0 \rangle$), formalized as a *dialectical tree* $\mathcal{T}_{\langle \mathcal{A}_0, Q_0 \rangle}$.

Example 8 *Consider $\langle \mathcal{A}_1, engine\_ok \rangle$ from Example 3, and the argumentation line shown in Example 7. Note that the argument $\langle \mathcal{A}_2, \sim fuel\_ok \rangle$ has a second (blocking) defeater $\langle \mathcal{A}_4, fuel\_ok \rangle$. The argument $\langle \mathcal{A}_1, engine\_ok \rangle$ has also a second defeater $\langle \mathcal{A}_5, \sim engine\_ok \rangle$. There are no more arguments to consider.*

There are three acceptable argumentation lines rooted in $\langle \mathcal{A}_1, engine\_ok \rangle$, namely:

$$\begin{aligned}
\lambda_1^{\langle \mathcal{A}_1, engine\_ok \rangle} &= [\ \langle \mathcal{A}_1, engine\_ok \rangle,\ \langle \mathcal{A}_2, \sim fuel\_ok \rangle,\ \langle \mathcal{A}_3, \sim low\_speed \rangle\ ] \\
\lambda_2^{\langle \mathcal{A}_1, engine\_ok \rangle} &= [\ \langle \mathcal{A}_1, engine\_ok \rangle,\ \langle \mathcal{A}_2, \sim fuel\_ok \rangle,\ \langle \mathcal{A}_4, fuel\_ok \rangle\ ] \\
\lambda_3^{\langle \mathcal{A}_1, engine\_ok \rangle} &= [\ \langle \mathcal{A}_1, engine\_ok \rangle,\ \langle \mathcal{A}_5, \sim engine\_ok \rangle\ ]
\end{aligned}$$

The corresponding dialectical tree $\mathcal{T}_{\langle \mathcal{A}_1, engine\_ok \rangle}$ rooted in the argument $\langle \mathcal{A}_1, engine\_ok \rangle$ is shown in Fig. 2.

Nodes in a dialectical tree $\mathcal{T}_{\langle \mathcal{A}_0, Q_0 \rangle}$ can be marked as *undefeated* and *defeated* nodes (U-nodes and D-nodes, resp.): all leaves in $\mathcal{T}_{\langle \mathcal{A}_0, Q_0 \rangle}$ will be marked U-nodes (as they have no defeaters), and every inner node is to be marked as *D-node* iff it has at least one U-node as a child, and as *U-node* otherwise. An argument $\langle \mathcal{A}_0, Q_0 \rangle$ is ultimately accepted as valid (or *warranted*) with respect to a DeLP program $\mathcal{P}$ iff the root of its associated dialectical tree $\mathcal{T}_{\langle \mathcal{A}_0, Q_0 \rangle}$ is labeled as *U-node*.

Example 9 *Consider the dialectical tree $\mathcal{T}_{\langle \mathcal{A}_1, engine\_ok \rangle}$ from Example 8. The marking procedure results in the nodes of $\mathcal{T}_{\langle \mathcal{A}_1, engine\_ok \rangle}$ marked as U-nodes and D-nodes as shown in Fig. 2.*[7]

Solving a query Q with respect to a given program $\mathcal{P}$ accounts for determining whether Q is supported by a warranted argument. Different doxastic attitudes are distinguished when answering query q according to the associated status of warrant, in particular:

(1) Yes: accounts for believing Q iff there is at least one warranted argument supporting Q on the basis of $\mathcal{P}$.
(2) No: accounts for believing $\sim Q$ iff there is at least one warranted argument

[7] The search space associated with dialectical trees is reduced by applying $\alpha - \beta$ pruning [26] (e.g., in Figure 2, if the right branch is computed first, then the left branch of the tree does not need to be computed).

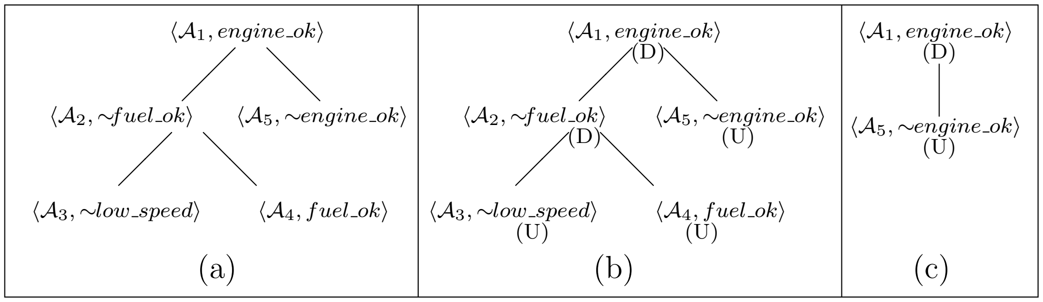

Fig. 2. (a) Dialectical tree for $\langle \mathcal{A}_1, engine\_ok \rangle$; (b) marked dialectical tree for $\langle \mathcal{A}_1, engine\_ok \rangle$; (c) associated pruned dialectical tree

supporting $\sim Q$ on the basis of $\mathcal{P}$.

(3) *Undecided*: neither Q nor $\sim Q$ are warranted wrt $\mathcal{P}$.

(4) *Unknown*: Q does not belong to the signature of $\mathcal{P}$.

Thus, according to DeLP semantics, given a program $\mathcal{P}$, solving a query Q will result in a value belonging to the set $Ans = \{Yes, No, Undecided, Unknown\}$.

Example 10 *Consider program $\mathcal{P}_{eng}$, and the goal engine_ok. The only argument supporting engine_ok is not warranted (as shown in Fig. 2). On the contrary, there exists an argument $\langle \mathcal{A}_5, \sim engine\_ok \rangle$ supporting $\sim engine\_ok$, and such argument has no defeaters, and therefore it is warranted. The answer to goal engine_ok will therefore be* No.
Consider now the same program $\mathcal{P}_{eng}$, and the goal $fuel\_ok$. The only argument supporting $fuel\_ok$ is $\langle \mathcal{A}_4, fuel\_ok \rangle$, which is defeated by a blocking defeater $\langle \mathcal{A}_2, \sim fuel\_ok \rangle$. The analysis for $\langle \mathcal{A}_2, \sim fuel\_ok \rangle$ is analogous, as this argument is defeated by $\langle \mathcal{A}_4, fuel\_ok \rangle$. Thus both arguments 'block' each other, neither of them being warranted. The resulting answer is Undecided.

4 Enhancing User Support System with Defeasible Argumentation

User support systems have had a steady evolution in the last decade. However, existing critic and recommender technologies are limited due to a number of reasons. On the one hand, they are incapable of dealing formally with the defeasible nature of users' preferences in complex environments. 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. On the other hand, they are not equipped with inference capabilities. As a consequence, much of the implicit information remains undiscovered. Quantitative approaches in AI, as opposed to qualitative approaches, have often been criticized for their inability to generate easy to understand and logically clean conclusions. The quantitative techniques adopted by most existing user support systems suffer from this limitation. The absence of an underlying formal model makes it hard to provide users with a clear explanation of the factors and procedures that led the system to come up with certain critics or recommendations. As a result, serious trustworthiness issues may arise, especially in those cases when business interests are involved, or when external manipulation is possible. Logic-based approaches could help to overcome these issues, enhancing recommendation technology by providing a means to formally express constrains and to draw inferences. However, as has been discussed by a number of sources, traditional logic-based systems are limited as they are unable to handle rules with exceptions, which are recurrent in recommendation scenarios. We contend that defeasible argumentation can be integrated into existing critics and recommenders, paving the way to tackle such problems.

Defeasible Logic Programming (DeLP) has proven to constitute a simple yet expressive language to encode rule-based knowledge with incomplete and potentially inconsistent information. Even though there are several alternative formalisms modelling argumentative reasoning [87], DeLP has shown to be a particularly useful alternative, as it provides a *multi-purpose programming language* based on a sound logical framework for default reasoning. In this context, DeLP has been successfully used in several real-world applications (including natural language processing [20], clustering [48], knowledge management [18,19], and multiagent systems [13], among others) In particular, recent research [47] has shown how DeLP can be also suitably extended as a scripting language for augmenting inference capabilities in the context of the Semantic Web [10].

Our proposal is to model users' preference criteria in terms of a DeLP program built on top of a traditional web search engine like Google. Figure 3 presents the basic architecture of a generic argument-based user support system based on DeLP. In this setting users preferences and background knowledge can be codified as facts and rules in a DeLP program. These facts and rules can come from different sources. For example, user's preferences could be entered explicitly by the user or could be inferred by the system (e.g., by monitoring the user's behavior.) Additional facts and rules could be obtained from other repositories of structured (e.g., databases) and semistructured data (e.g., the Web.)

We will distinguish particular subsets in a DeLP program, representing different elements in a user support system. For example, a DeLP program could take the form $\mathcal{P} = \mathcal{P}_{user} \cup \mathcal{P}_{pool} \cup \mathcal{P}_{domain}$, where sets $\mathcal{P}_{user}$ and $\mathcal{P}_{pool}$ represent preferences and behavior of the active user and the pool of users, respectively. In the case of the active user, his/her profile can be encoded as facts and rules in DeLP. In the case of the pool of users, rule induction techniques are in order [8] resulting in defeasible rules characterizing trends and general preference

[8] An approach for inducing defeasible rules from association rules can be found

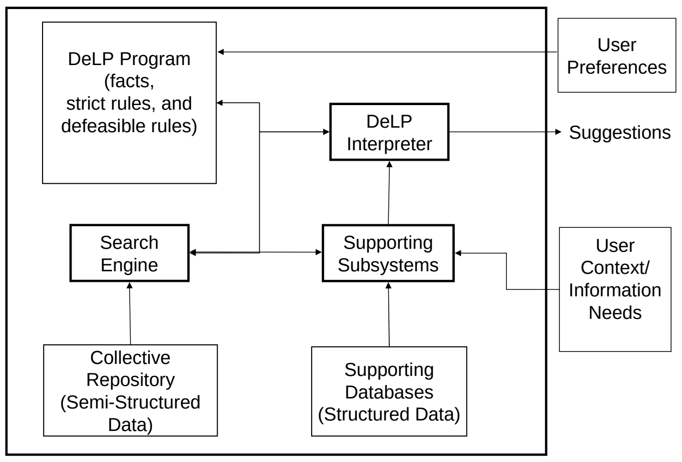
DeLP Program (facts, strict rules, and defeasible rules)
User Preferences
DeLP Interpreter
Suggestions
Search Engine
Supporting Subsystems
User Context/ Information Needs
Collective Repository (Semi-Structured Data)
Supporting Databases (Structured Data)

Fig. 3. Argument-Based User Support System based on DeLP

criteria (e.g., *normally if a given user likes X then she also likes Y*). The set $\mathcal{P}_{domain}$ represents the domain (background) knowledge, encoded using facts and rules in DeLP.

Either proactively or upon a user's request, an argument-based user support system triggers the search for suggestions. If needed, the collected results could be codified as facts and added to the DeLP program. Finally, a DeLP interpreter is in charge of performing the qualitative analysis on the program and to provide the final suggestions to the user.

Given the program $\mathcal{P}$, a user's request is transformed into suitable DeLP queries, from which different *suggestions* are obtained. For the sake of simplicity, we will assume in our analysis that user suggestions will be DeLP terms associated with a distinguished predicate name *rel* (which stands for *relevant* or *acceptable as a valid suggestion*). Using this formalization, suggestions will be classified into three sets, namely:

- S^w (warranted suggestions): those suggestions s_i for which there exists at least one warranted argument supporting $rel(s_i)$ based on $\mathcal{P}$.
- S^u (undecided suggestions): those suggestions s_i for which there is no warranted argument for $rel(s_i)$, neither there is a warranted argument for $\sim rel(s_i)$ on the basis of $\mathcal{P}$, and
- S^d (defeated suggestions): those suggestions s_i such that there is a warranted argument supporting $\sim rel(s_i)$ on the basis of $\mathcal{P}$.

Given a potential suggestion s_i, the existence of a warranted argument $\langle \mathcal{A}_1, rel(s_i) \rangle$ built on the basis of the DeLP program $\mathcal{P}$ will allow to conclude that s_i should be presented as a final suggestion to the user. If results are presented as a ranked list of suggestions, then a warranted suggestion is

in [50].

more relevant than an undecided one or defeated one. Note that the above classification has a direct correspondence with the doxastic attitudes associated with answers to DeLP queries. We must remark that it is always possible to ensure that the computation of warrant cannot lead to contradiction [44]: if there exists a warranted argument $\langle A, h \rangle$ on the basis of a program $\mathcal{P}$, then there is no warranted argument $\langle B, \sim h \rangle$ based on $\mathcal{P}$.

In the next sections we present two instances of this general framework: a web recommender and a word processing critic.

5 An Argument-Based Web Recommender

In this section we will outline an example (adapted from [29]) of how the proposed approach works in the context of solving news related queries. In this context, the recommendation system aims at providing an enriched web search engine which categorizes results. Thus, the resulting framework can be seen as a particular instance of an argument-based user support system, where the user's needs correspond to strings to be searched on the web. The search engine is a conventional search engine (e.g., GOOGLE). Final recommendation results for a query q are prioritized according to domain background knowledge and the user's declared preferences. Figure 4 illustrates the architecture of an argument-based news recommender system.

Given a user query q, it will be given as an input to a traditional content-based web search engine, returning a list of search results L=[s_1, s_2, s_3, s_4]. If required, the original query q could be suitably re-formulated in order to improve the quality of the search results to be obtained. In the list L we can assume that s_i is a unique name characterizing a piece of information $info(s_i)$, in which a number of associated features (meta-tags, filename, URL, etc.) can be identified. We assume that such features can be identified and extracted from $info(s_i)$ by some specialized tool, as suggested by Hunter [51] in his approach to dealing with structured news reports (see discussion in the last section). Such features will be encoded as a set $\mathcal{P}_{search}$ of new DeLP facts, extending thus the original program $\mathcal{P}$ into a new program $\mathcal{P}'$. A special operator **Revise** deals with possible inconsistencies found in $\mathcal{P}_{search}$ with respect to $\mathcal{P}'$, ensuring $\mathcal{P} \cup \mathcal{P}_{search}$ is not contradictory.[9] Following the algorithm shown in Fig. 5 we can now analyze s_1, s_2, s_3 and s_4 in the context of a new DeLP

[9] For example, contradictory facts may be found on the web; a simple belief revision criterion applied in an argument-based multiagent scenario [16] is to prefer the facts with a newer timestamp over the older ones. Among other possible criteria, preference could also be established regarding the reliability of the source of information.

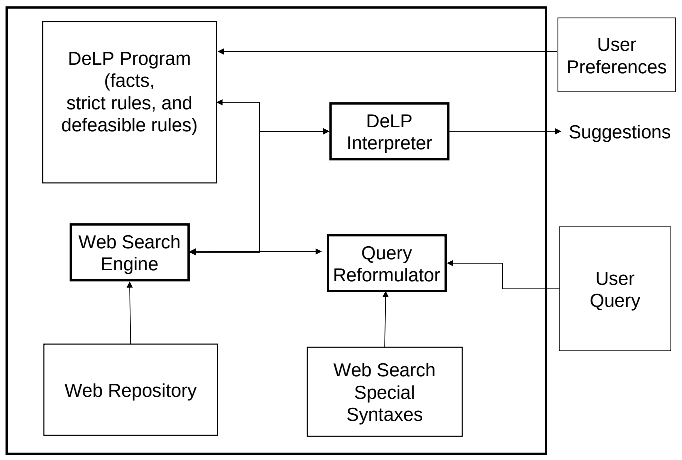

Fig. 4. Argument-Based Web Recommeder System based on DeLP

program $\mathcal{P}'=\mathcal{P} \cup Facts$, where $Facts$ denotes the set corresponding to the collection discussed above and $\mathcal{P}$ corresponds to domain knowledge and the user's preferences about the search domain.[10] For each s_i, the query $rel(s_i)$ will be analyzed in light of the new program $\mathcal{P}'$. Elements in the original list L of content-based search results will be classified into three sets of warranted, undecided, and defeated results, as discussed in Section 4. The final output presented to the user will be a sorted list L' in which the elements of L are ordered according to their epistemic status with respect to $\mathcal{P}'$. Figure 5 outlines a high level algorithm, which will be exemplified in the case study shown next.

5.1 Providing Recommendations for Web Search Queries: a Worked Example

Consider a journalist who wants to search for news articles about recent outbreaks of bird flu. A query q containing the terms *news*, *bird*, and *flu* will return thousands of search results. Our journalist may have some implicit knowledge to guide the search, such as: (1) she always considers relevant the newspaper reports written by Bob Beak; (2) she usually considers relevant the reports written by trustworthy journalists; (3) Reports written by trustworthy journalists which are out of date are usually not relevant; (4) Knowing that a journalist has not faked reports provides a tentative reason to believe he or she is trustworthy. By default, every journalist is assumed to be trustworthy. (5) Japanese and Thailandian newspapers usually offer a biased viewpoint on bird flu outbreaks; (6) The *"Japanese Times"* (http://jpt.jp) is a Japanese newspaper which she usually considers non biased; (7) Chin Yao Lin is known to have faked a report. Such rules and facts can be modelled in terms of a

[10] In this particular context, note that $\mathcal{P} = \mathcal{P}_{domain} \cup \mathcal{P}_{user}$.

```
ALGORITHM Recommend_on_Query
INPUT:  Query $q$, DeLP program $\mathcal{P} = \mathcal{P}_{user} \cup \mathcal{P}_{pool} \cup \mathcal{P}_{domain}$
OUTPUT:  List $L_{new}$ {recommendation results wrt $\mathcal{P}'$}
BEGIN
  Let $L = [s_1, s_2, \ldots s_k]$ be the output of solving query $q$
  wrt content-based search engine $SE$
  {$L$ is the list of (the first $k$) results obtained from query $q$ via $SE$ }
  $\mathcal{P}_{search}$ = {facts encoding $info(s_1)$, $info(s_2)$ ... $info(s_k)$}
  {$info(s_i)$ stands for features associated with result $s_i$ }
  $\mathcal{P}' :=$ Revise ($\mathcal{P} \cup \mathcal{P}_{search}$).
  {Revise stands for a belief revision operator to ensure consistency in $\mathcal{P}'$ }
  Initialize $S^w$, $S^u$, and $S^d$ as empty sets.
  {$S^w$, $S^u$, and $S^d$ stand for the set of results $s_i$'s which are warranted as
  relevant, undecided and warranted as non-relevant, respectively }
  FOR EVERY $s_i \in L$
    DO
      Solve query $rel(s_i)$ using DeLP program $\mathcal{P}'$
      IF $rel(s_i)$ is warranted THEN add $s_i$ to $S^w$
       ELSE
         IF $\sim rel(s_i)$ is warranted THEN add $s_i$ to $S^d$
           ELSE add $s_i$ to $S^d$
  Return Recommendation $L_{new} = [s^w_1, s^w_2, \ldots, s^w_{j1}, s^u_1, s^u_2, \ldots, s^u_{j2}, s^d_1, \ldots, s^d_{j3}]$
END
```

Fig. 5. Algorithm for solving queries in an argument-based web recommender system

```
              rel(X)  —<  author(X, A), trust(A).
             ~rel(X)  —<  author(X, A), trust(A), outdated(X).
            trust(A)  —<  not faked_news(A).
             ~rel(X)  —<  address(X, Url), biased(Url).
         biased(Url)  —<  thailandian(Url).
         biased(Url)  —<  japanese(Url).
        ~biased(Url)  —<  japanese(Url), domain(Url, D), D = "jpt.jp".
              rel(X)  ←   author(X, bob_beak).
          oudated(X)  ←   date(X, D), getdate(Today), (Today − D) > 100.
      thailandian(X)  ←   [Computed elsewhere]
         japanese(X)  ←   [Computed elsewhere]
      domain(Url, D)  ←   [Computed elsewhere]
          getdate(T)  ←   [Computed elsewhere]
faked_news(chin_yao_lin) ←
```

Fig. 6. DeLP program modelling preferences of a journalist

DeLP program $\mathcal{P}$ shown in Figs. 6 and 7.[11] Note that some rules in $\mathcal{P}$ rely on "built in" predicates computed elsewhere and not provided by the user.[12]

Consider the case for s_1. The search for an argument for $rel(s_1)$ returns $\langle \mathcal{A}_1, rel(s_1) \rangle$: that is, s_1 should be considered relevant since it corresponds

[11] Note that the predicate $biased(X)$ should be read as an abbreviation for "X is biased with regard to bird flu".

[12] For instance, the country of origin corresponding to a specific web domain can be found querying an Internet directory service such as WHOIS.

$author(s_1, chin\_yao\_lin).$ $address(s_1,$ "*jpt.jp/...*"$).$ $date(s_1, 20031003).$
$author(s_2, jen\_doe).$ $address(s_2,$ "*news.co.uk/...*"$).$ $date(s_1, 20001003).$
$author(s_3, jane\_truth).$ $address(s_3,$ "*jpt.jp/...*"$).$ $date(s_3, 20031003).$
$author(s_4, bob\_beak).$ $address(s_4,$ "*mynews.com/...*"$).$ $date(s_4, 20031003).$

Fig. 7. Facts encoded from original web search results

to a newspaper article written by Chin Yao Lin who is considered a trustworthy author (note that every journalist is considered to be trustworthy by default.) Here we have[13]

$$\begin{aligned}\mathcal{A}_1 \;=\; \{\; & rel(s_1) \prec author(s_1, chin\_yao\_lin), trust(chin\_yao\_lin)\;; \\ & trust(chin\_yao\_lin) \prec \mathsf{not}\;\; faked\_news(chin\_yao\_lin)\;\}.\end{aligned}$$

Search for defeaters for $\langle \mathcal{A}_1, rel(s_1)\rangle$ will result in finding a proper defeater $\langle \mathcal{A}_2, \sim rel(s_1)\rangle$: s_1 is not relevant as it comes from a Japanese newspaper, which is by default assumed to be biased about bird flu. In this case we have

$$\begin{aligned}\mathcal{A}_2 \;=\; \{\; & \sim rel(s_1) \prec address(s_1, \text{“}jpt.jp...\text{”}),\; biased\;(\text{“}jpt.jp...\text{”})\;; \\ & biased(\text{“}jpt.jp...\text{”}) \prec japanese\;(\text{“}jpt.jp...\text{”})\;\}.\end{aligned}$$

Note that we also have an argument $\langle \mathcal{A}_3, \sim biased(\text{“}jpt.jp...\text{”})\rangle$ which defeats $\langle \mathcal{A}_2, \sim rel(s_1)\rangle$, reinstating $\langle \mathcal{A}_1, rel(s_1)\rangle$: Usually articles from the "Japanese Times" are not biased. In this case we have

$$\begin{aligned}\mathcal{A}_3 \;=\; \{\; \sim biased(\text{“}jpt.jp...\text{”}) \prec\;\; & japanese(\text{“}jpt.jp...\text{”}), \\ & domain(\text{“}jpt.jp...\text{”}, \text{“}jpt.jp\text{”}), \\ & (\text{“}jpt.jp\text{”} \;=\; \text{“}jpt.jp\text{”})\;\}.\end{aligned}$$

Finally, another defeater for argument $\langle \mathcal{A}_1, rel(s_1)\rangle$ is found, namely the argument $\langle \mathcal{A}_4, faked\_news(chin\_yao\_lin)\rangle$, with $\mathcal{A}_4 = \emptyset$. No other arguments need to be considered. The resulting dialectical tree rooted in $\langle \mathcal{A}_1, rel(s_1)\rangle$ as well as its corresponding marking is shown in Figure 8a (left). The root node is a D-node (defeated), and hence $\langle \mathcal{A}_1, rel(s_1)\rangle$ is not warranted. Carrying out a similar analysis for $\sim rel(s_1)$ results in the dialectical tree shown in Figure 8a (right). The root node $\langle \mathcal{A}_2, \sim rel(s_1)\rangle$ is marked as D-node. There are no other candidate arguments to consider; hence s_1 is deemed as <u>*undecided*</u>.

[13] For the sake of clarity, we use semicolons to separate elements in an argument $\mathcal{A} = \{e_1\;;\; e_2\;;\; \ldots;\; e_k\;\}$.

The case of s_2 is analogous. The argument $\langle \mathcal{B}_1, rel(s_2) \rangle$ can be built, with

$$\begin{array}{rcl} \mathcal{B}_1 & = & \{\ rel(s_2) \prec author(s_2, jen\_doe),\ trust(jen\_doe); \\ & & trust(jen\_doe) \prec \mathsf{not}\ \ faked\_\ news\ (jen\_doe)\ \}. \end{array}$$

This argument is defeated by a proper defeater $\langle \mathcal{B}_2, {\sim}rel(s_2) \rangle$, with

$$\begin{array}{rcl} \mathcal{B}_2 & = & \{{\sim}rel(s_2) \prec author(s_2, jen\_doe),\ trust(jen\_doe),\ outdated(s_2)\ ; \\ & & trust(jen\_doe) \prec \mathsf{not}\ \ faked\_news(jen\_doe)\}. \end{array}$$

There are no more arguments to consider, and $\langle \mathcal{B}_1, rel(s_2) \rangle$ is deemed as non warranted (the resulting marked dialectical tree is shown in Fig. 8b (left)). The analysis of ${\sim}rel(s_2)$ results in an single argument. Consequently, its associated dialectical tree has a single node $\langle \mathcal{B}_2, {\sim}rel(s_2) \rangle$ and it is $\underline{warranted}$.

Following the same line of reasoning used in the case of s_1 we can analyze the case of s_3. An argument $\langle \mathcal{C}_1, rel(s_3) \rangle$, with

$$\begin{array}{rcl} \mathcal{C}_1 & = & \{\ rel(s_3) \prec author(s_3, jane\_truth),\ trust(jane\_truth)\ ; \\ & & trust(jane\_truth) \prec \mathsf{not}\ \ faked\_\ news(jane\_truth)\ \} \end{array}$$

can be built supporting the conclusion $rel(s_3)$ (a newspaper article written by Jane Truth is relevant as she can be assumed to be a trustworthy author). A defeater $\langle \mathcal{C}_2, {\sim}rel(s_3) \rangle$ will be found: s_1 is not relevant as it comes from a Japanese newspaper, which by default is assumed to be biased about bird flu. Here we have

$$\begin{array}{rcl} \mathcal{C}_2 & = & \{\ {\sim}rel(s_3) \prec address(s_3, \text{“}jpt.jp...\text{”}),\ biased\ (\text{“}jpt.jp...\text{”}); \\ & & biased(\text{“}jpt.jp...\text{”}) \prec japanese\ (\text{“}jpt.jp...\text{”})\ \}. \end{array}$$

But this defeater in its turn is defeated by a third argument $\langle \mathcal{C}_3,\ biased(s_3) \rangle$, as usually articles from the “Japanese Times” are not biased. In this case we have

$$\begin{array}{rll} \mathcal{C}_3 & = \{\ {\sim}biased(\text{“}jpt.jp...\text{”}) \prec & japanese(\text{“}jpt.jp...\text{”}), \\ & & domain(\text{“}jpt.jp...\text{”},\ \text{“}jpt.jp\text{”}), \\ & & (\text{“}jpt.jp\text{”}\ =\ \text{“}jpt.jp\text{”})\ \}. \end{array}$$

The resulting dialectical tree for $\langle \mathcal{C}_1, rel(s_3) \rangle$ is shown in Fig. 8c (left)). The original argument $\langle \mathcal{C}_1, rel(s_3) \rangle$ can be thus deemed as $\underline{warranted}$.

Finally let us consider the case of s_4. There is an argument $\langle \mathcal{D}_1, rel(s_4) \rangle$ with $\mathcal{D}_1 = \emptyset$, as $rel(s_4)$ follows directly from the strict knowledge in $\mathcal{P}$. Clearly, there is no defeater for an empty argument, and therefore $rel(s_4)$ is $\underline{warranted}$. The associated dialectical tree is shown in Fig. 8d.

Applying the criterion given in the algorithm shown in Fig. 5, the initial list

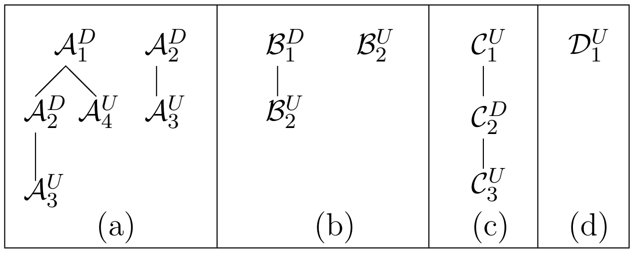

Fig. 8. Dialectical trees associated with (a) $\langle \mathcal{A}_1, rel(s_1)\rangle$ and $\langle \mathcal{A}_2, {\sim}rel(s_1)\rangle$; (b) $\langle \mathcal{B}_1, rel(s_2)\rangle$ and $\langle \mathcal{B}_2, {\sim}rel(s_2)\rangle$; (c) $\langle \mathcal{C}_1, rel(s_3)\rangle$ and (d) $\langle \mathcal{D}_1, rel(s_4)\rangle$

of search results $[s_1,\ s_2,\ s_3,\ s_4]$ will be shown as $[s_3,\ s_4,\ s_1,\ s_2]$ (as $\langle \mathcal{C}_1, rel(s_3)\rangle$ and $\langle \mathcal{D}_1, rel(s_4)\rangle$ are warranted, $\langle \mathcal{A}_1, rel(s_3)\rangle$ is undecided and $\langle \mathcal{B}_2, {\sim}rel(s_2)\rangle$ is warranted (i.e., s_2 is warranted to be a non-relevant result).

6 An Argument-Based Word Processing Critic

In this section we will show how the generic framework detailed in Section 4 can be used in the context of a word processing critic system. The goal of the proposed system is to provide proactive assistance for language usage assessment by combining web-based linguistic corpora and defeasible argumentation. Textual expressions are extracted from the user's document and evaluated with respect to usage indices, which are good indicators of the suitability of an expression on the basis of the current web corpus. The defeasible argumentation system determines if a given expression is acceptable by analyzing a defeasible logic program which encodes the user's preferences. Those expressions assessed as unsuitable are further inspected automatically by the system to help the user make the necessary repairs. Figure 9 illustrates the architecture of a word processing critic system as an instance of an argument-based user s

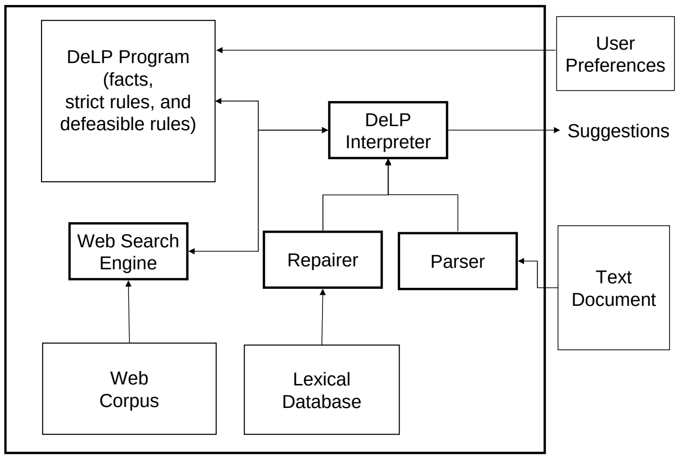

Fig. 9. Argument-Based Word Processing Critic System

A large sample of sentences in different natural languages has been accumulated as part of Web documents on the Internet. Most of such documents are accessible through search engines, whose pattern-matching capabilities have turned out to be useful for using the web space as a linguistic corpus, also called *Web-Corpus* [55]. Such a web corpus offers a number of advantages in comparison with traditional linguistic corpora [110]. In order to analyze relevant features of language usage patterns in web-based corpora, values associated with absolute or relative frequencies of string patterns wrt different web domains turn out to be particularly useful. We call such values *usage indices*. Such usage indices can be easily computed on the basis of advanced search facilities provided by most search engines (e.g. GOOGLE).

Next we introduce some definitions to formalize this concept. In the sequel, strings will be denoted with lowercase letters $s, t, u, \ldots$, possibly subscripted. We will use $d_1, d_2, \ldots$ to denote different web domains. The term *domain* will be used indistinctly to refer to complete web domain names (such as `'google.com'`) as to suffix portion of web domain names (e.g., `'.com'`). The distinguished constant name Web will be used to characterize the collection of all existing web domains. Given a domain d, we will use $\|d\|$ to denote the number of web pages found in the domain d. This notation can be extended to a set of domains $\mathcal{D} = \{d_1, d_2, \ldots, d_k\}$ as $\|\mathcal{D}\| = \sum_{i=1}^{k} \|d_i\|$. [14] Similarly, given a domain d and a string s, we will use $\|d\|_s$ to denote the number of web pages in domain d containing the string s. [15] Usage indices will be based on computing occurrences in sets of domains, as presented next.

Definition 11 (Usage indices U_g, U_c, U_r, and U_p) *Let s be a string, and let $\mathcal{D}$, $\mathcal{D}_1$ and $\mathcal{D}_2$ be non-empty sets of web domains, with $\mathcal{D} = \{d_1, d_2, \ldots, d_k\}$. We define the concepts of* general usage U_g*,* constrained usage U_c*,* ratio usage U_r*, and* prefix usage U_p *as follows:*

- $U_g(s) =_{def} \|Web\|_s$.
- $U_c(s, \mathcal{D}) =_{def} \|\mathcal{D}\|_s = \sum_{i=1}^{k} \|d_i\|_s$.
- $U_r(s, \mathcal{D}_1, \mathcal{D}_2) =_{def} ((U_c(s, \mathcal{D}_1) + 1)/(U_c(s, \mathcal{D}_2) + 1) \times (\|\mathcal{D}_2\|/\|\mathcal{D}_1\|)$.
- $U_p(s_1, s, \mathcal{D}) =_{def} U_c(s_1 \bullet s, \mathcal{D})/U_c(s, \mathcal{D})$ *if* $U_c(s, \mathcal{D}) \neq 0$*, and* 0 *otherwise.*

Given a string s, the constrained usage $U_c(s, \mathcal{D})$ represents the frequency of pages containing s restricted to the set $\mathcal{D}$ of web domains. The ratio usage $U_r(s, \mathcal{D}_1, \mathcal{D}_2)$ represents the ratio of the frequency of pages with s in $\mathcal{D}_1$ to the

[14] In the sequel, we will assume that domain names included in a domain set do not overlap, i.e., given a set of domains $\mathcal{D} = \{d_1, \ldots, d_k\}$ they satisfy that if $i \neq j$ then d_i is not a suffix domain of d_j. In addition, we will assume that all domains contain at least one web page.

[15] The special syntax *site:*, available in certain search engines (e.g., GOOGLE), restricts the search to a specified domain, allowing to obtain an estimation of $\|d\|_s$ and $\|d\|$ by posing the queries 's site:d' and 'site:d', respectively.

frequency of pages with s in $\mathcal{D}_2$. Finally, the prefix usage U_p informs about the likelihood of finding a string s_1 immediately preceding another string s in a page from some domain in $\mathcal{D}$.

Example 12 *Consider the strings s_1=rearing children, s_2=parents, and s_3=of twins. Let d_1=`'.uk'` and d_2=`'.babycentre.co.uk'`. Then it holds that*

$$
\begin{array}{lcl}
\|Web\| & = & 3307998701 \\
\|\{d_1\}\| & = & 28000000 \\
U_c(s_1, \{d_1\}) & = & 435 \\
U_c(s_1, Web) & = & 13700 \\
U_r(s_1, \{d_1\}, Web) & = & (436/13701) * (3307998701/28000000) = 3.76 \\
U_p(s_2, s_3, \{d_2\}) & = & 677/747 = 0.906.
\end{array}
$$

Note in the above example that statistical inference can be performed from usage indices (e.g., 90% of occurrences of the phrase of twins found in the URL `'.babycentre.co.uk'` are preceded by the word parents). Note also that the above computations are time-dependent (as they depend on the current Web corpus).[16]

Although the Web corpus provides very useful resources for language usage assessment on the basis of the relative and absolute frequencies in web documents, coming up with suggestions about language patterns requires a meta level analysis from the end user, who must perform an additional inference process based on such frequency values. Let us consider again the case of the journalist presented earlier, who thinks that a given expression E is not suitable for a news report intended for a Spanish newspaper, as he suspects that E is a regionalism e.g., from Argentina. This last assumption can be supported on the basis of the ratio $R = U_r(E, \{\text{`.ar'}\}, \{\text{`.es'}\})$. The fact that $R > 1$ provides a *tentative* reason for concluding that E is a regionalism associated with Argentina. Knowing that E is already in use in other Spanish newspaper may make the journalist change his mind, as he would have a reason that *defeats* the previous assumption. Once again, the above situation can be captured by computing $R' = U_c(E, \mathcal{D}_{news})$, where $\mathcal{D}_{news}$ corresponds to a set of domains corresponding to the Spanish mass media. The fact that $R' > \theta$, where θ is a particular threshold value, provides a reason to think that E is a common expression in the Spanish mass media, and therefore it can be used.

Our proposal aims at modelling the kind of analysis described above by integrating a front-end parser for the text entered by the user with a DeLP interpreter, which provides recommendations by solving queries on the basis of usage indices.

[16] Computations of usage indices in this article were performed using Google with the existing Web corpus on Feb. 19, 2004.

Given a text T corresponding to a user document, a front-end parser extracts a list $T' = [s_1, s_2, \ldots, s_k]$ of syntactic elements from T. Every $s_i \in T'$ is analyzed wrt a DeLP program $\mathcal{P}$, which encodes criteria for language usage in terms of strict and defeasible rules. Rules in $\mathcal{P}$ may include references to built-in predicates U_g, U_c, U_r and U_p which stand for usage indices as presented in Def. 11. A distinguished predicate name *solve* will be used for analyzing the *acceptability* of every expression s_i with respect to language usage criteria specified in $\mathcal{P}$. Program $\mathcal{P}$ contains the definition of a predicate called *acc*, which is used to evaluate the acceptability of its argument expression. Thus, the existence of a warranted argument $\langle A, acc(s_i)\rangle$ built on the basis of $\mathcal{P}$ will allow to conclude that s_i is an acceptable expression. Similarly, the existence of a warranted argument $\langle A, {\sim}acc(s_i)\rangle$ indicates that s_i is *not* acceptable.[17]

An interesting feature in automated systems for language assessment is the possibility of suggesting *repairs* whenever a particular user expression seems not suitable. This sort of functionality can be embedded in the proposed framework by means of a specialized predicate *repair*. Should an expression s_i be assessed as unacceptable, then *repair* can be used to seek for alternatives. An expression s_{new} is a potential repair for s_i if s_{new} is the result of replacing some words in s_i by synonyms found in a lexical database (e.g., WordNet [39]). If a warranted argument $\langle A, acc(s_{new})\rangle$ is built on the basis of $\mathcal{P}$, then s_{new} is presented to the user as a possible alternative to s_i. This process is outlined in the algorithm of Fig. 10.

6.1 An Example of Language Usage Assessment

Consider the case of an American journalist who writes articles in Spanish about Latinamerican issues, intended for audiences in Spain and Argentina. As Spanish is not his mother tongue, he usually makes mistakes related to properly assessing the correct language usage. A sample paragraph from such a journalist (and its corresponding English translation) could be as follows:

> *"El corralito fue un fenómeno muy complejo [...] Para el colectivo de los trabajadores autónomos cualquier liviano error tenía consecuencias [...]."*
> *"The "corralito"[18] was a very complex phenomenon [...] For the group of au-*

[17] Note that the *acc* predicate in this context plays a similar role to the *rel* predicate used in Section 5.

[18] The term "corralito" (little baby crib) was coined in Argentina in Dec. 2001 to denote severe restrictions on money drawing from banks due to an economic crisis in the country. The term became popular as mass media from different Spanish-speaking countries (including Spain) reported about the economic situation in Argentina, becoming hence an expression used to refer to an "abnormal situation in which customers are not allowed to draw their money from a bank for a long period

```
ALGORITHM ProvideAssessment
INPUT: Text $T$, DeLP program $\mathcal{P}$ modelling user preferences
OUTPUT: Assessment on $T$ {according to web corpus and $\mathcal{P}$}
        Suggest repairs when necessary
        {according to web corpus, lexical database and $\mathcal{P}$}
BEGIN
  Compute $T' = [s_1, s_2, \ldots s_k]$ on the basis of $T$
  {$T'$ results from parsing $T$. Every $s_i$ is a piece of text.}
  FOR EVERY $s_i \in T'$
    DO {try to solve acceptability of $s_i$}
      Solve query $acc(s_i)$ based on $\mathcal{P}$ and web corpus
      IF $acc(s_i)$ is warranted
      THEN Do nothing {assume $s_i$ is correct.}
      ELSE
         Solve query $\sim acc(s_i)$ based on $\mathcal{P}$ and web corpus
         IF $\sim acc(s_i)$ is warranted
        THEN {search for repairs}
            REPEAT
              Let $s'_i$ be a new candidate repair for $s_i$
              IF $acc(s'_i)$ is warranted
              THEN Suggest $s'_i$ as an alternative
            UNTIL (Repair $s'_i$ found) or (no more repairs available)
         ELSE {neither $acc(s_i)$ nor $\sim acc(s_i)$ holds}
            there is no suggestion about $s_i$
END
```

Fig. 10. Algorithm for providing language usage assessment using defeasible argumentation

*tonomous workers any *slight* mistake had consequences [...].”*

Let us assume that the editor of the newspaper will check every article written by our journalist before it is sent to print, guided by a number of criteria which characterize a “well-written document”. In the above text some anomalies will be detected: “corralito” is a common term in Argentina, but not so common in Spain (except in the news). The expression “colectivo de trabajadores autónomos” (group of autonomous workers) [43] has a clear meaning in Spain, but is not understood in Argentina (as “agrupación” or “grupo” is the Argentinean equivalent for “colectivo”). The noun phrase “*liviano error” is wrong in Spanish language, as the correct fixed idiom would be “ligero error” (=slight mistake), even though the adjectives “ligero” and “liviano” are accepted as synonyms in this context. Some of the possible criteria the editor could apply to avoid such anomalies can be characterized in terms of the DeLP program shown in Fig. 11.

of time”.

Control rules for language usage assessment:

1) $solve(S) \leftarrow acc(S),\ write(\text{‘Acceptable’}).$

2) $solve(S) \leftarrow {\sim}acc(S), repair(S,R),$
$acc(R),\ write(\text{‘Acc. if rephrased as’}, R).$

3) $solve(S) \leftarrow {\sim}acc(S),\ write(\text{‘Not acceptable’}).$

4) $solve(\_) \leftarrow write(\text{‘Undecided. No suggestion found’}).$

5) $repair(S,R) \leftarrow simple\_nphrase(S), S = [Noun, Adj],$
$syn(Adj, NAdj), R = [Noun, NAdj].$

6) $syn(Adj, NAdj) \leftarrow list\_syn(Adj, L),\ member(NAdj, L).$

Defeasible rules capturing language usage preferences:

7) $acc(S) \;-\!\!\!\prec\; common\_in\_spanish(S).$

8) ${\sim}acc(S) \;-\!\!\!\prec\; rare\_in\_spanish(S).$

9) ${\sim}acc(S) \;-\!\!\!\prec\; common\_in\_spanish(S),$
$regionalism(S, [\texttt{‘.ar’}]).$

10) ${\sim}acc(S) \;-\!\!\!\prec\; common\_in\_spanish(S),$
$regionalism(S, [\texttt{‘.es’}]).$

11) $regionalism(S, Ctry) \;-\!\!\!\prec\; locally\_freq(S, Ctry).$

12) ${\sim}regionalism(S, [\texttt{‘.ar’}]) \;-\!\!\!\prec\; locally\_freq(S, Ctry),$
$appears\_in\_news(S, \texttt{‘.es’}).$

Predicates based on computing Usage Indices:

13) $common\_in\_spanish(S) \leftarrow spanish\_speaking(Cs),$
$V \text{ is } U_c(S, Cs),\ V > 200.$

14) $rare\_in\_spanish(S) \leftarrow$ not $common\_in\_spanish(S).$

15) $appears\_in\_news(S, C) \leftarrow news\_domains(Ds, C),$
$V \text{ is } U_c(S, Ds), V > 200.$

16) $locally\_freq(S, [\texttt{‘.ar’}]) \leftarrow V \text{ is } U_r(S, C, [\texttt{‘.es’}]), V > 10$

17) $locally\_freq(S, [\texttt{‘.es’}]) \leftarrow V \text{ is } U_r(S, C, [\texttt{‘.ar’}]), V > 10$

Additional predicates:

18) $news\_domains([\texttt{‘elmundo.es’}, \texttt{‘elpais.es’}], \texttt{‘.es’}).$

19) $spanish\_speaking([\texttt{‘.es’}, \texttt{‘.ar’}).$

20) $list\_syn(\text{liviano}, [\text{ligero}, \text{sutil}, ...]).$

21) $member(X, [X|\_]).$

22) $member(X, [Y|Z]) \leftarrow member(X, Z).$

23) $simple\_nphrase(S) \leftarrow$[computed elsewhere].

Fig. 11. DeLP program modelling preference criteria for acceptable language usage patterns in newspaper articles

Rules 1 to 4 characterize the behavior of the *solve* predicate as outlined before. Rule 5 defines the *repair* predicate restricted to simple noun phrases of the form $[Noun, Adj]$. Repairs consist in just replacing *Adj* by an alternative synonym obtained from an ad-hoc predicate *syn* (Rule 6).[19] For the sake of simplicity, in this example the definition of synonym is restricted to the Span-

[19] A lexical database such as WordNet [39] can provide a list of synonyms (synset) for an arbitrary adjective.

ish adjective liviano ("light"). Defeasible rules 7 to 12 capture language usage preferences on the basis of usage indices computed in rules 13 to 15. Rule 7 establishes that strings whose general frequency in Spanish speaking countries is above a certain threshold value are defeasibly acceptable. From Rule 8 it follows that strings which cannot be proven to be common in web domains from Spanish speaking countries are usually not acceptable. Rules 9 and 10 establish that regionalisms from Argentina and Spain are usually not acceptable. Rule 11 specifies when a given expression can be defeasibly assumed to be a regionalism in terms of its frequency, computed using the $locally\_freq$ predicate. Rule 12 provides an exception for the above rule: a string S which is locally frequent in Argentina but is also frequent in the Spanish media is not considered to be a regionalism. A string s is considered frequent in the Spanish media if a considerable percentage of all the hits found for s in Spain are found in newspapers. Rule 18 lists some news domains and rule 19 specifies that the Spanish speaking countries to be considered for the analysis are Spain and Argentina.[20]

Suppose we now apply the high-level algorithm presented in Fig. 10, where the strings extracted from the above text are s_1, s_2 and s_3, with s_1=corralito, s_2=colectivo de los trabajadores autónomos, and s_3=liviano error. Consider the case for string s_1. The search for a warranted argument for $acc(s_1)$ returns $\langle \mathcal{A}_1, acc(s_1)\rangle$, with

$$\mathcal{A}_1 = \{\ acc(s_1) \prec common\_in\_spanish(s_1)\ \}$$

This argument holds since $U_c(s_1,[\text{`.es'},\text{`.ar'}]) > 200$.The DeLP inference engine will then search for defeaters for $\langle \mathcal{A}_1, acc(s_1)\rangle$. A proper defeater $\langle \mathcal{A}_2, \sim acc(s_1)\rangle$ is found: s_1 is not acceptable as there are reasons to think it is a regionalism from Argentina. Here we have

$$\begin{array}{rcl} \mathcal{A}_2 & = & \{\ \sim acc(s_1) \prec common\_in\_spanish(s_1),\ regionalism(s_1, \text{'.ar'})\ ; \\ & & regionalism(s_1, \text{'.ar'}) \prec locally\_freq(s_1, \text{'.ar'})\}. \end{array}$$

Note that $\langle \mathcal{A}_2, \sim acc(s_1)\rangle$ is a proper defeater for $\langle \mathcal{A}_1, acc(s_1)\rangle$ as the first argument is based on more specific information than the second. Note also that predicate $locally\_freq(s_1, \text{`.ar'})$ holds, as $U_r(s_1,[\text{`.ar'}],[\text{`.es'}]) = 33.1 > 10$. A defeater for this argument can be found on its turn: corralito is not a regionalism in Argentina as it is fairly frequent in the Spanish news. Here we have the argument

$$\begin{array}{rcll} \mathcal{A}_3 & = & \{\ \sim regionalism(s_1, [\text{`.ar'}]) \prec & locally\_freq(s_1, \text{`.ar'}), \\ & & & appears\_in\_news(s_1, \text{`.es'})\}. \end{array}$$

[20] For the sake of simplicity, in this example we restrict our analysis to only these two countries (Spain and Argentina), and we only focus on exceptions for regionalisms in Argentina based on sample Spanish news domains.

Note that predicate $appears\_in\_news(s_1, \text{spain})$ holds, as $U_c(\textsf{corralito}, \mathcal{D})$=40, with $\mathcal{D}$ representing domains from Spanish newspapers. Note also that the definition of dialectical tree does not allow the use of $\langle \mathcal{A}_1, acc(s_1)\rangle$ to defeat again $\langle \mathcal{A}_2, \sim acc(s_1)\rangle$, as this would imply falling into *fallacious*, circular argumentation. After the above analysis no other defeater can be found. The resulting dialectical tree rooted in $\langle \mathcal{A}_1, acc(s_1)\rangle$ as well as its corresponding marking is shown in Fig. 12a. The root node is marked as U-node (undefeated), which implies that the argument $\langle \mathcal{A}_1, acc(s_1)\rangle$ is warranted.

Consider now the case for string s_2=colectivo de los trabajadores autónomos. There is an argument $\langle \mathcal{B}_1, acc(s_2)\rangle$, with

$$\mathcal{B}_1 = \{acc(s_2) \prec common\_in\_spanish(s_2)\ \}$$

which holds following the same reasoning as above. However, there is a defeater for $\langle \mathcal{B}_1, acc(s_2)\rangle$, namely $\langle \mathcal{B}_2, \sim acc(s_2)\rangle$, with

$$\begin{array}{rcl} \mathcal{B}_2 & = & \{\ \sim acc(s_2) \prec common\_in\_spanish(s_2),\ regionalism(s_2, [\texttt{`.es'}]); \\ & & regionalism(s_2, \texttt{`.es'}) \prec locally\_freq(s_2, \texttt{`.es'})\}. \end{array}$$

As above, predicate $locally\_freq(s_2, \texttt{`.es'})$ holds, as $U_r(s_2, [\texttt{`.es'}], [\texttt{`.ar'}])$= 41.4. No other arguments can be computed from here onwards. The *solve* predicate will thus fire the search for a warranted argument for $\sim acc(s_2)$, which is successful (a dialectical tree rooted in $\langle \mathcal{B}_2, \sim acc(s_2)\rangle$ with no defeaters). The resulting situation is shown in Fig. 12b. Note that no repair is possible here, as *repair* is only for simple noun phrases.

Finally, let us consider the case for the string s_3=liviano error. There is no argument (and consequently no warranted argument) supporting the conclusion $acc(s_3)$, as $common\_in\_spanish(s_3)$ does not hold: s_3 is syntactically correct but is pragmatically wrong as noun phrase in Spanish. In contrast, there is a warranted argument $\langle \mathcal{C}_1, \sim acc(s_3)\rangle$ which provides a reason *not* to accept s_3, based on rule 8, with

$$\mathcal{C}_1 = \{\ \sim acc(s_3) \prec rare\_in\_spanish(s_3)\}.$$

The predicate *solve* will try to repair s_3, obtaining a new alternative string s'_3 = ligero error, searching then for a warranted argument for $acc(s'_3)$. A warranted argument for concluding $acc(s'_3)$ can be found, namely the argument $\mathcal{D}_1$={ $acc(s'_3) \prec common\_in\_spanish(s'_3)$}. As a side effect, the warning message *"Accepted if rephrased as* ligero error*"* will be given to the user. This situation is shown in Fig. 12c.

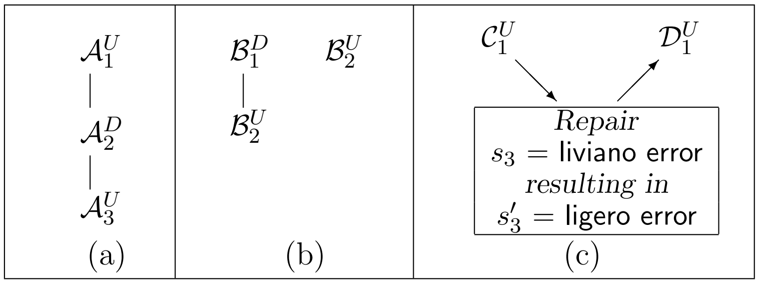

Fig. 12. Dialectical trees associated with (a) $\langle \mathcal{A}_1, acc(s_1) \rangle$ (b) $\langle \mathcal{B}_1, acc(s_2) \rangle$ and $\langle \mathcal{B}_2, {\sim}acc(s_2) \rangle$; (c) $\langle \mathcal{C}_1, {\sim}acc(s_3) \rangle$ and $\langle \mathcal{D}_1, acc(s'_3) \rangle$

7 Implementation issues. Ongoing work

Performing defeasible argumentation is a computationally complex task. An abstract machine called JAM (Justification Abstract Machine) has been specially developed for an efficient implementation of DeLP [44]. JAM provides an argument-based extension of the traditional WAM (Warren's Abstract Machine) for Prolog. A full-fledged implementation of DeLP is freely available online[21], including facilities for visualizing arguments and dialectical trees. A Java-based Integrated Development Environment (IDE) for DeLP has also been developed [101]. This Java version of DeLP allows to compile DeLP code into JAM opcodes. A visual environment for interacting with DeLP programs is provided. Several features leading to efficient implementations of DeLP have also been recently studied, in particular those related to comparing conflicting arguments by specificity [102], computing dialectical trees efficiently [27] and extending DeLP to incorporate possibilistic reasoning [30,25,31]. Equivalence results with other extensions of logic programming have also been established [28].

In this article we have presented and examined two instances of argument-based user support systems: a web recommender systems and a critiquing system. The results returned by the system were encouraging for several analyzed cases. Some preliminary experiments were performed on the basis of a prototype version implemented using the existing Java-based DeLP environment [101]. However, it must be remarked that these initial experiments only serve as a "proof of concept" prototype. As part of our future work we expect to perform more thorough evaluations of such prototypes. These will include automatic evaluations of argument-based recommendations using publicly available datasets (e.g., the MovieLens dataset [72].) A subset of a selected dataset will be used for training purposes while the remaining will be used for testing the system performance. These datasets usually contain users' ratings of objects such as movies, books, music, etc. In many cases, objects and users are associated with a series of properties (e.g, user's age, user's occu-

[21] See `http://lidia.cs.uns.edu.ar/DeLP`

pation, movie's genre, etc.) Properties associated with users and objects in the training set will be mined to extract defeasible rules of the form "users who like objects with property X usually like objects with property Y." We expect to apply standard automatic evaluation techniques such as "leave-one-out" cross-validation to estimate the accuracy of the system recommendations.

While new recommendation technologies can be systematically evaluated by means of available datasets, the performance of critiquing techniques is hard to assess in an automatic way. In the future we expect to carry out user studies to test whether users can complete certain tasks (e.g., translating a document from English to Spanish) more effectively with the support of an argument-based critic system. Measures of effectiveness will be defined in terms of product quality and amount of time spent on the task. It must also be remarked that in our current proposal there is no hierarchical distinction between personal preferences and pool preferences. As a matter of future work we are analyzing the possibility of envisioning a more evolved framework with an order on preferences: an unexperienced user could mostly benefit from preferences coming from the users' pool whereas a more experienced user could give a higher priority to his own preferences.

As discussed in Section 5, the general framework for an argument-based recommender system includes the possibility of inducing defeasible rules automatically. A first approach for inducing defeasible rules from association rules [79,1] can be found in [50]. A natural extension for considering induced defeasible rules in the context of our proposal is to have weights attached to such rules on the basis of the existing evidence. In this respect, recent research has led to the development of P-DeLP [30,25] an extension of DeLP which includes the possibility of representing such weights and performing inferences with them. Part of our ongoing work is related to extending the current framework with P-DeLP in order to have weighted defeasible rules, which can be more easily obtained by using association rules techniques.

Finally, one aspect which deserves particular attention is the comparison between DeLP knowledge encoding capabilities and other rule-based systems, which have the well-known disadvantage of being brittle [46]: adding or deleting one rule may substantially change the behavior of the system, possibly harming transparency and thus trustworthiness. In this respect, DeLP inherits all declarative features from logic programming, and as such is highly elaboration tolerant. Answers are always supported by arguments built on the basis of a given DeLP program. Clearly, it is possible that a change in the program (such as the addition or deletion of some program rules) might change the answer obtained as an output for a given query, but any of such changes can be perfectly traced back and justified by showing the underlying dialectical analysis that led to this change.

8 Related work

Several kinds of user support systems that operate on top of Internet services have been proposed over the past years (e.g., [3,74,62,78]). In the case of web-based recommender systems, the usual approach involves taking into account the user's interests –either declared by the user or conjectured by the system– to rank or filter web pages.

Some web recommender systems include LIRA [7], BASAR [105], ifWeb [5], SOAP [108], Let's Browse [63], SurfLen [42], Margin Notes [96], and Quickstep [71]. An example of hybrid news filtering system is NewsDude [12], a learning agent that is trained by the user with a set of interesting news articles. A hybrid social chat recommender system is Butterfly [106], a system that uses keywords to find interesting conversations in Usenet newsgroups. Collaborative news recommender systems include GroupLens [92,57] and PHOAKS [103]. These recommender systems, however, differ from our proposal in that they do not attempt to perform a qualitative analysis to warrant recommendations.

Recently there has been a growing attention to the development of a Semantic web [9,10], and a number of semantically enhanced recommendation techniques have been proposed (e.g., [111]). Although the vision of the Semantic Web is still at its beginning, the use of defeasible reasoning for qualitative analysis can also be naturally integrated into such approaches, as recently shown in [8].

Providing assessment in word-processing activities has long been a source of research in the natural language processing community [59]. In this context, the term *critiquing system* refers to cooperative tools that observe the user interacting with a word-processing tool and present reasoned opinions about the user-entered text, helping to discover and point out errors that might otherwise remain unnoticed. Most popular word-processing critiquing systems include spelling-, grammar-, and style-checkers [32]. Many current approaches to word-processing critiquing systems aim at the analysis of language usage patterns on the basis of the rich source of textual material that the Web offers as a linguistic corpus [91,110]. Several concordancers and writing assistant tools were developed. WebLEAP [110] is a software which aims to inform people of the existence and the popularity of expressions in the Web. It gets an expression as input and returns the frequencies in Web documents of the word sequences extracted from the given expression by using a search engine. The expression and the frequencies are presented graphically so that the user can estimate the result easily and judge with more confidence. WebCorp [91] is a suite of tools which allows access to the World Wide Web as a corpus. WebCorp is designed to retrieve linguistic data from the Web: concordance lines showing the context in which the user's search term occurs. Finally, KWiCFinder [41] is an utility for conducting online searches without supervision, facilitating more finely-targeted searches than typical web search engines. KWiCFinder returns "Key Word in Context" (KWiC) abstracts and highlights the search terms, so that the user can evaluate the usefulness of documents matching the query. All the above systems provide information on language usage on the basis of frequency values computed from the Web corpus, including also advanced facilities for restricting search to particular domains and finding grammatical patterns. However, differently from our proposal, these systems leave the analysis of the returned language expressions to the user.

9 Conclusions

We have presented a a novel approach towards the development of user support systems, by enhancing recommendation technologies through the use of qualitative analysis using argumentation. We have shown how DeLP provides a suitable tool for carrying on such analysis. Our approach operates on top of a conventional search engine such as Google, providing thus a tool for higher abstraction when dealing with users' information needs.

One important issue in our proposal is the need to extract relevant features from Web search results, encoding them as part of a DeLP program. Although HTML tags associated with Web documents are not intended to convey a formal semantics, these tags can be usefully exploited to extract meaningful content [35,4,60]. On the other hand, the emergence of XML as a standard for data representation on the Web contributes to further simplify the above problem. In this context, the approach proposed by Hunter [51–53] to represent semi-structured text through logical formulas is particularly relevant for enhancing the capabilities of argument-based user support systems as presented in this article. We think that in future developments this process could be complemented by additional techniques, such as defeasible rule discovery [50] and specialized argument assistance tools [107].

Current trends in user support system technologies show that in the coming years we can expect the development of very efficient environments for decision making, where both quantitative and qualitative analysis of user preferences will play important roles. In this context, we contend that the integration of defeasible argumentation techniques will provide a powerful tool which will contribute to help fulfill this goal.

Acknowledgements

The authors would like to thank anonymous reviewers for their valuable suggestions and comments which helped improve the original version of this article. This research work was supported by Projects TIC2003-00950, TIN 2004-07933-C03-03, by Ramón y Cajal Program (MCyT, Spain) by CONICET (Argentina), and by Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 No. 13.096).

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD Conference*, pages 207–216, 1993.

[2] L. Amgoud, N. Maudet, and S. Parsons. An argumentation-based semantics for agent communication languages. In *Proc. of the 15th. European Conference in Artificial (ECAI), Lyon, France*, 2002.

[3] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell. Webwatcher: A learning apprentice for the world wide web. In *AAAI Spring Symp. on Information Gathering*, pages 6–12, 1995.

[4] N. Ashish and C. A. Knoblock. Semi-automatic wrapper generation for internet information sources. In *Conference on Cooperative Information Systems*, pages 160–169, 1997.

[5] F. Asnicar and C. Tasso. ifWeb: a prototype of user models based intelligent agent for document filtering and navigation in the World Wide Web. In *Sixth International Conference on User Modeling*, Chia Laguna, Sardinia, Italy, June 1997.

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

[7] M. Balabanovic, Y. Shoham, and Y. Yun. An adaptive agent for automated web browsing. *Journal of Visual Communication and Image Representation*, 6(4), 1995.

[8] N. Bassiliades, G. Antoniou, and I. P. Vlahavas. A defeasible logic reasoner for the semantic web. In *Proc. of the Workshop on Rules and Rule Markup Languages for the Semantic Web*, pages 49–64, 2004.

[9] T. Berners-Lee. Semantic web road map. Technical report, W3C Design Issues, 1998.

[10] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[11] P. Besnard and A. Hunter. A logic-based theory of deductive arguments. *Artif. Intell.*, 128(1-2):203–235, 2001.

[12] D. Billsus and M. J. Pazzani. A hybrid user model for news classification. In *In Kay J. (ed.), UM99 User Modeling - Proceedings of the Seventh International Conference*, pages 99–108. Springer-Verlag, 1999.

[13] R. Brena, C. Chesñevar, and J. Aguirre. Argumentation-supported information distribution in a multiagent system for knowledge management. In *Proc. of the Second Intl. Workshop on Argumentation in Multiagent Systems (ArgMAS 2005), Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2005), Utrecht, Netherlands*, pages 13–29, July 2005.

[14] J. Budzik and K. J. Hammond. User interactions with everyday applications as context for just-in-time information access. In *Proceedings of the 2000 International Conference on Intelligent User Interfaces*, New Orleans, Louisiana, 2000. ACM Press.

[15] J. Budzik, K. J. Hammond, and L. Birnbaum. Information access in context. *Knowledge based systems*, 14(1–2):37–53, 2001.

[16] M. Capobianco, C. Chesñevar, and G. Simari. An argument-based framework to model an agent's beliefs in a dynamic environment. *Lecture Notes in Artificial Intelligence (Proc. of the First International Workshop on Argumentation in Multiagent Systems. AAMAS 2004 Conference, New York, USA)*, 3366:96–111, 2005.

[17] D. Carbogim, D. Robertson, and J. Lee. Argument-based applications to knowledge engineering. *The Knowledge Engineering Review*, 15(2):119–149, 2000.

[18] C. Chesñevar, R. Brena, and J. Aguirre. Knowledge distribution in large organizations using defeasible logic programming. In *Proc. of the 18th Canadian Conference on AI (published in LNCS, Vol. 3501, Springer Verlag)*, pages 244–256, 2005.

[19] C. Chesñevar, R. Brena, and J. Aguirre. Modelling power and trust for knowledge distribution: an argumentative approach. In *Proc. of the 3rd Mexican International Conference on Artificial Intelligence (MICAI 2005). To appear in LNCS Springer Series (in press)*, Nov. 2005.

[20] C. Chesñevar and A. Maguitman. An Argumentative Approach to Assessing Natural Language Usage based on the Web Corpus. In *Proc. of the ECAI-2004 Conference. Valencia, Spain*, pages 581–585, Aug. 2004.

[21] C. Chesñevar and A. Maguitman. ArgueNet: An Argument-Based Recommender System for Solving Web Search Queries. In *Proc. of the 2nd IEEE Intl. IS-2004 Conference. Varna, Bulgaria*, pages 282–287, June 2004.

[22] C. Chesñevar, A. Maguitman, and R. Loui. Logical Models of Argument. *ACM Computing Surveys*, 32(4):337–383, Dec. 2000.

[23] C. Chesñevar, A. Maguitman, and G. Simari. A first approach to argument-based recommender systems based on defeasible logic programming. In *Proc. of the 10th Intl. Workshop on Non-Monotonic Reasoning (NMR-2004). Whistler, Canada*, pages 109–117, June 2004.

[24] C. Chesñevar and G. Simari. Towards computational models of natural argument using labelled deductive systems. In C. Reed, editor, *Proc. of the 5th Intl.. Workshop on Computational Models of Natural Argument (CMNA 2005), 19th Intl. Joint Conf. in Artificial Intelligence (IJCAI 2005). Edimburgh, UK*, pages 32–39, July 2005.

[25] C. Chesñevar, G. Simari, T. Alsinet, and L. Godo. Modelling agent reasoning in a logic programming framework for possibilistic argumentation. In *Proc. of 2nd European Workshop on Multiagent Systems. Barcelona, Spain*, pages 135–142, Dec. 2004.

[26] C. Chesñevar, G. Simari, and A. García. Pruning Search Space in Defeasible Argumentation. In *Proc. of the Workshop on Advances and Trends in AI*, pages 46–55. XX Intl. Conf. of the SCCC, Santiago, Chile, Nov. 2000.

[27] C. Chesñevar, G. Simari, and L. Godo. Computing dialectical trees efficiently in possibilistic defeasible logic programming. *Lecture Notes in Artificial Intelligence LNAI 3662 (Proc. of the 8th Intl. Conference on Logic Programming and Nonmonotonic Reasoning LPNMR)*, pages 158–171, Sept. 2005.

[28] C. I. Chesñevar, J. Dix, F. Stolzenburg, and G. R. Simari. Relating Defeasible and Normal Logic Programming through Transformation Properties. *Theoretical Computer Science*, 290(1):499–529, 2003.

[29] C. I. Chesñevar and A. G. Maguitman. Combining argumentation and web search technology: Towards a qualitative approach for ranking results. *Intl. Journal of Advanced Computational Intelligence & Intelligent Informatics*, 9(1):53–60, 2005.

[30] C. I. Chesñevar, G. Simari, T. Alsinet, and L. Godo. A Logic Programming Framework for Possibilistic Argumentation with Vague Knowledge. In *Proc. of the Intl. Conference in Uncertainty in Artificial Intelligence (UAI 2004). Banff, Canada*, pages 76–84, July 2004.

[31] C. I. Chesñevar, G. Simari, L. Godo, and T. Alsinet. Argument-based expansion operators in possibilistic defeasible logic programming: Characterization and logical properties. *Lecture Notes in Artificial Intelligence (LNAI) 3571 (Proc. of the 8th European Conference on Symbolic and Qualitative Aspects of Reasoning with Uncertainty – ECSQARU 2005, Barcelona, Spain)*, pages 353–365, July 2005.

[32] K. Church and L. Rau. Commercial applications of natural language processing. *CACM*, 38(11):71–79, November 1995.

[33] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *In Proceedings of ACM SIGIR Workshop on Recommender Systems*, August 1999.

[34] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[35] R. B. Doorenbos, O. Etzioni, and D. Weld. A scalable comparison-shopping agent for the world-wide web. In W. L. Johnson and B. Hayes-Roth, editors, *Procs. of the First Intl. Conf. on Autonomous Agents (Agents'97)*, pages 39–48, Marina del Rey, CA, USA, 1997. ACM Press.

[36] P. M. Dung. On the Acceptability of Arguments and its Fundamental Role in Nomonotonic Reasoning and Logic Programming. In *Proc. of the 13th. International Joint Conference in Artificial Intelligence (IJCAI), Chambéry, Francia*, pages 321–357, 1993.

[37] D. Engelbart. Augmenting human intellect: A conceptual framework. Summary report, Stanford Research Institute, on Contract AF 49(638)-1024, October 1962.

[38] D. M. D. N. P. P.-S. e. F. Baader, D. Calvanese. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, 2003.

[39] C. Fellbaum. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

[40] G. Fischer, K. Nakakoji, J. Ostwald, G. Stahl, and T. Sumner. Embedding computer-based critics in the contexts of design. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 157–164. ACM Press, 1993.

[41] W. Fletcher. Concordancing the Web with KWiCFinder. In *Proc. 3rd North American Symposium on Corpus Linguistics and Language Teaching.* American Assoc. for Applied Corpus Linguistics, 2001.

[42] X. Fu, J. Budzik, and K. J. Hammond. Mining navigation history for recommendation. In *Intelligent User Interfaces*, pages 106–112, 2000.

[43] B. Galimberti and R. Russell, editors. *The Oxford Spanish Dictionary.* Oxford University Press, 2003.

[44] A. García and G. Simari. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004. Preliminary version available from `cs.uns.edu.ar/∼grs`.

[45] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *Proceedings of the 7th International Conference on Logic Programming. Jerusalem*, June 1990.

[46] J. Giarratano and G. Riley. *Expert Systems: Principles and Programming.* PWS Publishing Company, 3rd. edition, 1998.

[47] S. Gómez, C.Chesñevar, and G.Simari. Incorporating defeasible knowledge and argumentative reasoning in web-based forms. In *Proc. of the 3rd Intl. Workshop on Intelligent Techniques for Web Personalization (ITWP 2005). 19th Intl. Joint Conf. in Artificial Intelligence (IJCAI 2005). Edimburgh, UK*, pages 9–16, July 2005.

[48] S. Gómez and C. Chesñevar. A Hybrid Approach to Pattern Classification Using Neural Networks and Defeasible Argumentation. In *Proc. of 17th Intl. FLAIRS Conference. Miami, Florida, USA*, pages 393–398. American Association for Artificial Intelligence, May 2004.

[49] T. Gordon and N. Karacapilidis. The Zeno Argumentation framework. In *Proc. of the Int. Conf. on Artificial Intelligence and Law (ICAIL-97), Melbourne, Australia*, pages 10–18, 1997.

[50] G. Governatori and A. Stranieri. Towards the application of association rules for defeasible rules discovery. In B. V. et. al, editor, *Legal Knowledge and Information Systems*, pages 63–75, Amsterdam, 2001. JURIX, IOS Press.

[51] A. Hunter. Hybrid argumentation systems for structured news reports. *Knowledge Engineering Review*, pages 295–329, 2001.

[52] A. Hunter. Logical fusion rules for merging structured news reports. *Data and Knowledge Engineering*, 42:23–56, 2002.

[53] A. Hunter. Merging structured text using temporal knowledge. *Data and Knowledge Engineering*, 41:29–66, 2002.

[54] A. C. Kakas and F. Toni. Computing argumentation in logic programming. *Journal of Logic and Computation*, 9(4):515–562, 1999.

[55] A. Kilgarriff. Web as Corpus. In *Proc. Corpus Linguistic Conf.*, pages 342–344. UCREL-Lancaster Univ, UK, 2001.

[56] J. A. Konstan. Introduction to recommender systems: Algorithms and evaluation. *ACM Trans. Inf. Syst.*, 22(1):1–4, 2004.

[57] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[58] R. Kowalski and F. Toni. Abstract Argumentation. *Artificial Intelligence and Law*, 4(3-4):275–296, 1996.

[59] K. Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439, 1992.

[60] N. Kushmerick, D. S. Weld, and R. B. Doorenbos. Wrapper induction for information extraction. In *IJCAI'97*, pages 729–737, 1997.

[61] D. B. Leake, T. Bauer, A. Maguitman, and D. C. Wilson. Capture, storage and reuse of lessons about information resources: Supporting task-based information search. In *Proceedings of the AAAI-00 Workshop on Intelligent Lessons Learned Systems. Austin, Texas*, pages 33–37. AAAI Press, 2000.

[62] H. Lieberman. Letizia: An agent that assists web browsing. In *IJCAI'95*, pages 924–929. Morgan Kaufmann, 1995.

[63] H. Lieberman, N. W. V. Dyke, and A. S. Vivacqua. Let's browse: a collaborative Web browsing agent. In *Proceedings of the 1999 International Conference on Intelligent User Interfaces (IUI'99)*, pages 65–68, Los Angeles, CA, USA, 1999. ACM Press.

[64] V. Lifschitz. Foundations of logic programming. In *Principles of Knowledge Representation*, pages 69–127. CSLI Publications, 1996.

[65] F. Linton, D. Joy, and H.-P. Schaefer. Building user and expert models by long-term observation of application usage. In *Proceedings of the seventh international conference on User modeling*, pages 129–138. Springer-Verlag New York, Inc., 1999.

[66] R. Loui, P. Norman, J. Altepeter, D. Pinkard, D. Craven, J. Lindsay, and M. Foltz. Progress on Room 5: A testbed for public interactive semi-formal legal argumentation. In *Proc. of the Int. Conf. on Artificial Intelligence and Law (ICAIL-97), Melbourne, Australia*, pages 207–214, 1997.

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

[68] N. Maudet and D. J. Moore. Dialogue games for computer supported collaborative argumentation. In *Proceedings of the 1st Workshop on Computer supported collaborative argumentation (CSCA99)*, Standford, USA, 1999.

[69] J. McCarthy. *Formalizing Common Sense: Papers by John McCarthy.* Ablex Publishing Corporation, 1990.

[70] J. McCarthy and P. J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[71] S. Middleton, D. DeRoure, and N. Shadbolt. Capturing knowledge of user preferences: Ontologies in recommender systems. In *Proceedings of the ACM K-CAP'01*, Victoria, Canada, 2001. ACM Press.

[72] B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266, New York, NY, USA, 2003. ACM Press.

[73] P. Miller. Attending: Critiquing a physician's management plan. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI-5)*, pages 449–461, September 1983.

[74] D. Mladenic. Personal webwatcher: Design and implementation. Technical report ijs-dp-7472, School of Computer Science, Carnegie-Mellon University, Pittsburgh, USA, 1996.

[75] D. Nute. Defeasible Reasoning. In J. H. Fetzer, editor, *Aspects of Artificial Intelligence*, pages 251–288. Kluwer Academic Publishers, Norwell, MA, 1988.

[76] S. Parsons and P. McBurney. Argumentation-based Dialogues for Agent Coordination. *Group Decision and Negotiation (to appear)*, 2004.

[77] S. Parsons, C. Sierrra, and N. Jennings. Agents that Reason and Negotiate by Arguing. *Journal of Logic and Computation*, 8:261–292, 1998.

[78] M. J. Pazzani, J. Muramatsu, and D. Billsus. Syskill & webert: Identifying interesting web sites. In *AAAI/IAAI'96*, pages 54–61, 1996.

[79] G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In G. Piatetsky-Shapiro and W. Frawley, editors, *Knowledge Discovery in Databases*. AAAI/MIT Press, Cambridge, MA, 1991.

[80] J. Pollock. *Knowledge and Justification*. Princeton, 1974.

[81] J. L. Pollock. Defeasible Reasoning. *Cognitive Science*, 11:481–518, 1987.

[82] J. L. Pollock. *Cognitive Carpentry: a blueprint for how to build a person*. Bradford/MIT Press, 1995.

[83] D. Poole. Explanation and Prediction: an Architecture for Default and Abductive Reasoning. *Computational Intelligence*, 5:97–110, 1989.

[84] D. L. Poole. On the Comparison of Theories: Preferring the Most Specific Explanation. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 144–147. IJCAI, 1985.

[85] H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-classical Logics*, 7:25–75, 1997.

[86] H. Prakken and G. Sartor. The role of logic in computational models of legal argument - a critical survey. In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond*, pages 342–380. Springer, 2002.

[87] H. Prakken and G. Vreeswijk. Logical Systems for Defeasible Argumentation. In D. Gabbay and F.Guenther, editors, *Handbook of Phil. Logic*, pages 219–318. Kluwer, 2002.

[88] F. Ramírez Bustamante and F. Sánches León. GramCheck: A grammar and style checker. In *Proceedings of the International Conference on Computational Linguistics (COLING-96)*, pages 175–181, 1996.

[89] C. Reed and D. Walton. Applications of argumentation schemes. In *Conference of the Ontario Society for the Study of Argument (OSSA2001), Windsor, Canada*, 2001.

[90] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13(1,2):81–132, Apr. 1980.

[91] A. Renouf. Webcorp: providing a renewable data source for corpus linguists. In G. et.al, editor, *Extending the scope of corpus-based research: new applications, new challenges.*, pages 219–318. Rodolpi, 2002.

[92] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[93] P. Resnick and H. R. Varian. Recommender systems. *CACM*, 40(3):56–58, 1997.

[94] B. Rhodes and P. Maes. Just-in-time information retrieval agents. *IBM Systems Journal special issue on the MIT Media Laboratory*, 39(3-4):685–704, 2000.

[95] B. J. Rhodes. *Just-In-Time Information Retrieval.* PhD thesis, MIT Media Lab, May 2000.

[96] B. J. Rhodes. Margin notes: Building a contextually aware associative memory. In *The Proceedings of the International Conference on Intelligent User Interfaces (IUI '00)*, January 2000.

[97] T. Selker. COACH: a teaching agent that learns. *Communications of the ACM*, 37(7):92–99, 1994.

[98] B. G. Silverman. Survey of expert critiquing systems: practical and theoretical frontiers. *Communications of the ACM*, 35(4):106–127, 1992.

[99] G. Simari and R. Loui. A Mathematical Treatment of Defeasible Reasoning and its Implementation. *Artificial Intelligence*, 53:125–157, 1992.

[100] G. R. Simari, A. J. García, and M. Capobianco. Actions, planning and defeasible reasoning. In *10th International Workshop on Non-Monotonic Reasoning (NMR 2004), Whistler, Canada*, pages 377–384, 2004.

[101] A. Stankevicius, A. Garcia, and G. Simari. Compilation techniques for defeasible logic programs. In *Proc. of the 6th Intl. Congress on Informatics Engineering*, pages 1530–1541. Univ. de Buenos Aires, Bs. Aires, Argentina, Ed. Fiuba, Apr. 2000.

[102] F. Stolzenburg, A. García, C. Chesñevar, and G. Simari. Computing Generalized Specificity. *Journal of Non-Classical Logics*, 13(1):87–113, 2003.

[103] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. PHOAKS: a system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, 1997.

[104] L. G. Terveen and D. A. Wroblewski. A collaborative interface for editing large knowledge bases. In *Proceedings of AAAI*, pages 491–496, Boston, 1990.

[105] C. G. Thomas and G. Fischer. Using agents to personalize the Web. In *Proceedings of the 2nd international conference on Intelligent user interfaces*, pages 53–60. ACM Press, 1997.

[106] N. W. Van Dyke, H. Lieberman, and P. Maes. Butterfly: a conversation-finding agent for internet relay chat. In *Proceedings of the 4th international conference on Intelligent user interfaces*, pages 39–41. ACM Press, 1999.

[107] B. Verheij. Artificial argument assistants for defeasible argumentation. *Artificial Intelligence Journal*, 150:291–324, 2003.

[108] A. Voss and T. Kreifelts. SOAP: social agents providing people with useful information. In *Proceedings of the international ACM SIGGROUP conference on Supporting group work : the integration challenge*, pages 291–298. ACM Press, 1997.

[109] G. Vreeswijk. Abstract argumentation systems. *Artif. Intell.*, 90(1-2):225–279, 1997.

[110] T. Yamanoue, T. Minami, I. Ruxton, and W. Sakurai. Learning Usage of English KWICly with WebLEAP/DSR. In *Proc. 2nd. Intl. Conf. on Inf. Technology and Applications (ICITA-2004) (to appear)*, 2004.

[111] C.-N. Ziegler. Semantic web recommender systems. In *Proceedings of the Joint ICDE/EDBT Ph.D. Workshop*, 2004.